Received 23 October 2022; accepted 2 November 2022. Date of publication 7 November 2022;
date of current version 5 December 2022. The review of this article was coordinated by Editor Lina Mohjazi.

Digital Object Identifier 10.1109/OJVT.2022.3219898

Data-Driven Malware Detection for 6G Networks: A Survey From the Perspective of Continuous Learning and Explainability via Visualisation

DILARA T. UYSAL [1], PAUL D. YOO [1] (Senior Member, IEEE), AND KAMAL TAHA [2] (Senior Member, IEEE)

[1]Department of Computer Science and Information Systems, Birkbeck College, University of London, WC1E 7HX London, U.K.
[2]Centre for Cyber-Physical Systems (C2PS), Khalifa University, Abu Dhabi 127788, UAE

CORRESPONDING AUTHOR: Paul D. Yoo (e-mail: p.yoo@bbk.ac.uk)

ABSTRACT 5G is inherently prone to security vulnerabilities. We witness that many today's networks contain 5G security flaws due to their reliance on the existing 4G network core. A lack of security standards for 5G IoT devices means network breaches and malware threats might run uncontrolled. The future 6G network is predicted to be implemented with artificial intelligence-driven communication via machine learning, enhanced edge computing, post-quantum cryptography and so forth. With the activation of edge computing, the computing power available at supercomputing servers is to be integrated directly into the devices at the entry point of a network in a distributed manner (*e.g.*, antennas, routers, IoT sensors, *etc*). This feature brings an equal quality of service everywhere including remote regions (*a.k.a* service everywhere) which will trigger an exponential growth of associated applications. In this intricate environment, malware attacks are becoming more challenging to detect. This paper thus reviews the theoretical and experimental data-driven malware detection literature, in the large-scale data-intensive field, relating to: (1) continuous learning, including new concepts in multi-domain to multi-target learning and the challenges associated with unseen/unknown data, imbalance data and data scarcity, and (2) new explainability via visualisation concepts with a multi-labelling approach which allows identifying malware by their recipes while improving the interpretability of its decision process.

INDEX TERMS Malware detection, dynamic/hybrid/static analysis, malware images, segmentation, machine learning, continuous machine learning and explainability.

I. INTRODUCTION

Today, it is no surprise that cyber-risk is top priority for businesses, governments and societies across the globe. It is widely known that malware is one of the greatest security threats today's enterprises face. The most recent US Treasury report [1] states that suspicious ransomware-related transactions in the US alone were worth $590 million in the first half of the year 2021. Although there has been great effort in the development of malware detection/prevention techniques adopting various cutting-edge technologies, sophisticated malware variants have been developed using such technologies.

It is evident that we will witness the networked connection of people, process, data and things (*a.k.a* Internet of Everything or IoE) in the near future. This would mean that 'everything' will soon be a potential target for cyber adversaries. With the deployment of more and more 5G networks, the security concerns of the connected 5G physical things have largely been reported which triggered the exploratory study of 6G networks. In the foreseeable future, 6G will make these linked things evolve into the connected intelligence via real-time/distributed intelligence edge computing. With 6G, it is anticipated that the today's service-based architecture will evolve into 'service everywhere' (SE) which will dramatically increase the number of associated applications. To deal such exponential growth, the organisational issues of frequency/spectrum allocation and memory access for interoperability among the many different services will need to be resolved. In such complex SE environment, however, detecting malware will be more challenging. Security services, for example, should be able to quickly plug into and detach from a huge number of microservices in distributed computing nodes. Attacks of various dimensions and network technologies necessitate a more sophisticated defence mechanism [72].

As a result of the integration of information technology (IT) systems with operational technology (OT) systems in 6G networks, it is anticipated that OT network sensors could have zero-day vulnerabilities while Supervisory Control and Data Acquisition (SCADA) systems may have vulnerabilities and endpoints may be targeted by malware, ransomware, DDoS, and spear fishing [84]. Many 6G conventional risks include DDoS, malware injection, and side channel attacks, which could be traced all the way back to prior generations. Meanwhile, the rapid spread of Artificial Intelligence (AI) exposes the 6G network to new threats manipulating AI's availability and integrity [74]. Learning on various edge devices introduces 6G networks to distributed AI vulnerabilities such as data poisoning. A malicious user could insert a few correctly labeled, minimally perturbed samples into the training set thereby affecting its ability to produce accurate predictions although deep probabilistic machine learning has recently demonstrated its potential in detecting such attacks by leveraging uncertainty estimates [94].

As the number of IoT-enabled devices in 6G era is expected to grow significantly, more and more malwares targeting such devices will prevail. IoT generally refers to dynamic and ever-evolving environments generating high-volume streams of heterogeneous data, in that complex, abnormal patterns could evolve considerably as time passes. As a result, we envision that new security mechanisms must be capable of detecting unknown malware variants while rapidly adapting to changes over time. However, conventional machine-learning-based methods are in isolation. If a change occurs in its feature-space distribution, the model needs to be re-built. Lifelong machine learning (LML) could assist in overcoming such limitation as its core is built on a continuous learning concept. At the same time, the development of a multi-label solution using machine learning, identifying malwares by their recipes or DNA is another under-researched area in the literature. This will not only provide transparency in decision making but also bring faster adoption by building user trust.

Nowadays, there is a greater need to develop accurate, flexible, scalable, and transparent data-driven methods to cope with future demands that are in line with similar 6G security-related initiatives. Such data-oriented modelling is important for a number of cyber defence and security areas, as they affect many related industries. Despite the fact that there is a demand for such data-driven and intelligent methods for the new large and complex data-intensive fields, very few of these literatures have addressed the recent and promising concepts such as continuous learning, federated/transfer learning, zero-/few-shot, multimodal learning and Explainable Artificial Intelligence (XAI) approaches [2], [3].

This paper, hence, provides a comprehensive review of the state-of-the-art data-driven literatures in malware detection including theoretical, empirical, and experimental studies pertaining to the various needs and recommendations. The objective of this paper is to introduce a new perspective for engineers, scientists and researchers in security, communications, and computer science domains, as well as to provide its roadmap for future research endeavours. To the best of our knowledge, this is the first study that investigates the data-driven malware detection problem from the perspective of continuous, explainable, and visualisable machine-learning-based strategy for creating a next-level malware solution for 6G networks.

This paper is organised as follows. Section II discusses the current developments in malware data, features and algorithms pointing out known/unknown issues. Section III examines the suitability of the continuous learning while discussing the limitations of current machine-learning-based methods. Section IV gives recommendations from a new explainability via visibility perspective. Section V concludes the paper.

II. A ROADMAP

A. DATASETS

In data-driven malware detection, data does matter. Choosing the right data to learn models from can make or break your analysis. Firstly, the volume of data relating to data updateness is an important criterion. VirusTotal [6], VirusShare [7], and MalwareBazaar [8] contain nearly all known malware, and they are continuously growing in terms of size as newly detected ones are added. EMBER [9] is another large dataset which has over 1 million records, however, the samples were collected in 2018 only. VX Heaven [10] and DREBIN [11] released before 2015, are out of date.

Secondly, if the target class of data has an uneven distribution of observations (*i.e.*, imbalance), then it may not be an accurate representation of the population, or it can discard useful information about the data itself which could be necessary for learning models. To learn an accurate malware detector, using a balanced data for malware and benign, as well as for each malware family is of great importance.

Malimg dataset [5] contains 9,339 malware images from 25 families and seems to be suitable for multi-class categorisation. Ahmed et al. [86] developed a CNNs architecture with the Malimg dataset for classifying malware in 5G-enabled Industrial Internet of Things (IIoT). There is also no requirement for pre-processing with Malimg for image-based analysis, however, in terms of number of observations, malware families are significantly imbalanced. DREBIN contains 5,560 malware samples out of 12K in total. Although it is still imbalanced, Comodo Cloud Security Centre [13] has an equal amount of malicious and non-malicious samples with 22,500 malware and 22,500 benign raw files. MalwareBazaar hold a greater diversity of known malware families and number of samples across categories is well balanced. MalwareBazaar also gives family labels.

Thirdly, to build a quality dataset, we need to identify raw data and add one or more meaningful and informative labels in order to provide context that a model could be learned from. Adding a label to each observation which determines whether its malicious or not and which family it belongs to is time-consuming and costly. To detect malicious Windows portable executable (PE) formatted files, EMBER provides a publicly labelled dataset of 900K training samples (300K malicious, 300K benign, 300K unlabelled) along with 200K test samples. These are either collected privately or via VirusTotal and VirusShare. However, the raw binaries are not provided, and as a result, it is not feasible to apply deep learning or extract new features from these executables [14]. Microsoft's Malware Classification Challenge [12] announced in 2015 provides a rich dataset of labelled and unlabelled samples from nine different malware types. It contains over 20K malware samples' disassembly and bytecode although its raw data is not publicly available. The malicious files of VirusShare are not labelled – no information on which family they are belong to. While VirusTotal API [15] which provides antivirus engines to label samples is widely used today. Rhode et al. [16] collected malicious and benign files from VirusTotal and other free software websites. When new malicious samples were found, the labels were verified with VirusTotal. All other samples were, however, labelled as benign. This, thus, does not seem to be a reliable method for labelling unseen malwares.

Some of the most popular applications available in Google Play Store collect benign samples. Unfortunately, this does not guarantee that there is no malware sample within the collected samples. While VirusShare makes their malicious samples publicly available, legitimate binaries cannot be shared freely due to copyright restrictions.

All in all, having a sufficient number of up-to-date samples for each malware class is of great importance. This allows detecting newly released malware attacks with machine learning which is notoriously data hungry. As previously noted, MalwareBazaar contains labelled samples. This could be combined with VirusShare and VirusTotal which are being constantly updated, in order to have a sufficient collection of well-balanced and up-to-date malware samples.

B. FEATURES

To detect malware, structural and behavioural features of files are examined. This allows understanding of the intent of malware. There are primarily three different approaches for extracting features, namely, static, dynamic and hybrid. Static analysis examines PE files without executing samples. It is fast and straightforward, yet, prone to code obfuscation. Dynamic analysis which runs malware in a virtual environment can avoid such obfuscation techniques. It, however, is time-consuming and can be deceived by anti-virtualisation techniques. Meanwhile, hybrid analysis which combines static and dynamic analysis has gained attention recently. Some recent studies [17], [18], [19] have implemented hybrid analysis and demonstrated a great performance throughout the detection process. Rafiq et al. [85] suggested an adversarial-based evasion approach to defend against evasion attempts. To prevent malicious adversaries from evading detection in IIoT with the combination of a 5G/6G network, a hybrid strategy was used. This employed dynamic features, system call features, and static features, as well as android intents and permissions for malware detection. Details of these analysis approaches along with their feature extraction methods and extracted features are summarised in Table 1.

TABLE 1. Feature Extraction

Analysis Type	Extraction Methods	Extracted Features
Static	Assembly code based	Opcodes
	PE structure based	Imports, exports, strings, DLLs, CFG, API
Dynamic	System resource based	CPU, memory usage
	Machine activity based	File, registry, network, process
	Function call based	System calls, API calls

1) STATIC ANALYSIS

Static analysis essentially examines files without executing them. With this analysis, it is difficult to predict a file's behaviour. It is, however, possible to approximate the actions it may eventually perform [20]. Static analysis extracts the features to obtain insight into the intent and structure of a malware (*e.g.*, strings, headers and opcodes).

Strings hold essential semantic data and reveal produced and modified filenames in addition to registry-related information. Strings may also reveal IP addresses used by malware authors for communication, URLs for command-and-control centres, malware author and group signatures, as well as group memberships. By analysing these data, the intent of a malware could be identified. Changing all interpretable strings is impractical in majority of programmes regardless of the type of malware created through recompilation or obfuscation techniques. Strings feature is, thus, regarded as one of the most important features for malware detection [21].

A PE file comprises many sections and headers that assist the dynamic linker on how to map the file into memory. Researchers often use header information where multiple fields holding structural information such as API import/export tables, references to dynamic libraries, several file sections, and type of metadata [20]. It is important to be able to process the header information in order to locate and disassemble the binary code that is commonly utilised in static analysis [22]. Wang et al. [23] used the PE header to identify several types of malware and they concluded that PE headers' entries are helpful for detecting virus and email worms. Although the detection performance could be maintained or improved with a small number of features from headers, they determined that they were insufficient for detecting trojan and backdoor. Manavi et al. [24] used the images generated from PE header to detect ransomware with a CNN model. Using the PE header, they achieved a detection accuracy of 93%. Due to its destructive effects, ransomware differs structurally from benign files and the header images could demonstrate this difference. The header of an executable file gives a useful structural information. Balram et al. [25] analysed the detection performance using strings feature and PE header feature. Their experimental results confirmed that string features are more informative than PE header features.

Machine language instruction consists of opcodes that is an indicator of the operation executed by CPU which can be used as a feature for testing opcode frequency or analysing similarities between opcode sequences [14]. Haddadpajouh et al. [83] developed a new malware detection method on cloud edge gateway devices to identify malware at an early stage and successfully decreased the damage caused by it. The model was trained using IoT malware samples' Opcode and Bytecode. Malware samples developed with the same source code or belonging to the same family share a substantial number of instruction blocks/segments. The frequency distribution of malware opcodes appears to deviate greatly from that of non-malicious software as rarer opcodes appear to explain greater variance in frequency than common opcodes [21]. Although the opcode feature can be beneficial for malware classification, it is limited by its incapability to define the invocation connections between instructions [26].

Static analysis provides a variety of structural information that assists in understanding the behaviour of programmes. In addition, it assures many additional information presenting samples such as export table, file resource information and file property [5]. Although static analysis is a fast analysis tool, obfuscation techniques could make it unreliable.

2) DYNAMIC ANALYSIS

Dynamic analysis examines a programme's behaviour by executing it in an isolated environment (*e.g.*, virtual machine). In contrast to static analysis, it can monitor what malware does in real time, and extract information every step. Consequently, dynamic analysis produces reliable findings. It can also identify any modifications made to the code while avoiding obfuscation techniques.

Function calls are used by the programmes for various purposes. An application programming interface (API) gives a way for computer programmes to communicate with each other (*e.g.*, between software and operating system). The behaviour of the application can be examined by tracing API or system calls meaning malware's suspicious behaviour can also be identified. The feature of API functions allows users to keep track of critical functions and API/system calls can be extracted via both static and dynamic analysis. However, several faulty API calls in the executable header inserted to confuse malware analysts could cause static analysis to be inefficient. Ding et al. [27] suggested an association mining approach for detecting malware based on API call traces. They saved time and cost by improving the rule quality and using different API selection criteria. Salehi et al. [28] introduced a malware detection approach based on APIs. They observed that a feature set combining API names, API return values, and/or input arguments increases the performance of a model that uses return values, and API calls features in the identification of unseen behaviour. When return value was used in the feature set with API calls and their parameters, a specific behaviour model and robust results was produced.

API function calls are used by programmes to perform various operations associated with networks, registry keys and files [17]. Kakisim et al. [29] experimentally proved that API calls, use of system libraries, and operation sequences are critical features for malware analysis.

Registry key provides important information about downloaded software, hardware, values, and options operated by processes. Dynamic analysis extracts info on registry modifications (*e.g.*, written/opened registry) are utilised by malware to avoid detection by security systems such as firewalls.

File system activity, on the other hand, extracts info on changes of a file such as deletion, creation and modification. Mohaisen et al. [30] suggested a new method for malware classification and automated labelling based on such behavioural characteristics. This method uses extracted features from memory, file system, registry and network activities. Han et al. [31] examined the outcomes of the models with various feature profiles including fundamental structural features, low-level behavioural characteristics (API calls and DLLs), and high-level behavioural characteristics are formed from files, registries, and network-behaviours. This high-level behaviour-based model outperformed other approaches. Their experiment demonstrated that the detection accuracy of using the combination of basic-structure and low-level behaviour profiles reached the peak.

Malware can download, upload, or configure data through the internet. By monitoring network traffic, associated IP addresses, HTTPs and DNS requests and URLs could be collected. These features are useful in comprehending the malicious intent. Malware needs to communicate with a remote server over the internet in order to carry out malicious actions. For example, most spam botnets are connected with their command and control (C&C) server using HTTP [32].

Dynamic analysis is a popular method as is thought to be more effective than static analysis. This is due to the fact that it can analyse programmes without having to disassemble them. Dynamic analysis can also detect both known and unknown malware [21]. Nevertheless, it is a slow method that requires more relevant resources.

3) HYBRID ANALYSIS

It is well-referenced that runtime-packed malware cannot be detected via static analysis. Dynamic techniques may not identify all malware capabilities. For this reason, static analysis is an essential complement to dynamic analysis. Hybrid analysis combines static and dynamic characteristics to benefit from the advantages of both. Eskandari et al. [33] proposed a new hybrid-based malware detection technique. In their learning phase, static and dynamic analysis were merged, however, only static analysis was employed in the scoring phase. This hybrid approach not only improved the detection performance but also shorten the testing process.

Ma et al. [34] proposed an ensemble malware classifier with the goal of reducing false positives. They employed the integrated features and the import functions extracted statically and dynamically and trained with a support vector machine and a classification and regression tree. Both classifier outputs were combined into a single final output. They concluded that their ensemble model enhanced their detection performance by decreasing false positives. Hadiprakoso et al. [17] observed that adding dynamic features to static features can improve the accuracy of their android malware detection model by 5%. Shijo and Salim [19] compared the results of static, dynamic, and hybrid analysis. The accuracies of their static analysis using the PCI feature and dynamic analysis using the API calls feature were 95.8% and 97.1%, respectively. Their accuracy increased to 98.7% when the static feature vector and dynamic feature vector were combined.

C. ALGORITHMS

Traditional machine-learning-based approaches for malware detection have utilised Random Forest [77], Navies Bayes [79], Decision Trees [80], SVM [30], and Logistic Regression [25]. In a study of [78], the performance of classifiers such as k-Nearest Neighbour (kNN), Naive Bayes, Decision Tree, Support Vector Machine (SVM), and Multi-Layered Perceptron (MLP) were compared for malware detection. J48-based decision tree demonstrated the highest accuracy (96.8%) based on this comparison of all experimental data from the 5 classifiers while providing a human-understandable rules. One of the greatest challenges with learning models using traditional machine-learning algorithms is the process of feature engineering. Deep learning circumvents these challenges as it is capable of extracting features in an automated fashion. Without much guidance from the programmer, deep learning algorithms focus on the right features by themselves. Although deep learning algorithms are classified as a supervised learner, unsupervised learners perform greatly in detecting unknown/unseen malwares.

1) UNSUPERVISED LEARNING

Unsupervised learning finds patterns in data without labels. In malware detection, autoencoder has widely been used to reduce the data dimension and to identify unknown threats. The bottleneck layer of an autoencoder could give a compressed representation of its input data. Its decoder layer reconstructs the data from the encoding representation in an unsupervised fashion.

Yousefi-Azar et al. [36] proposed a feature learning framework utilising an autoencoder for cybersecurity tasks. Their study not only generated a small number of latent features that capture semantic similarities between feature vectors but also learned semantic similarity from input features automatically. By reducing the dimension of the features, the memory requirement was subsequently reduced. Naway and Li [37] introduced a new android malware detection approach in which an autoencoder is utilised for classifying malwares rather than feature reduction. Their autoencoder was modified to learn in a semi-supervised fashion.

Restricted boltzmann machine (RBM) is an algorithm for describing all training samples in a more meaningful and compact manner by capturing their internal structure and regularities [41]. Its goal is to reduce the error between the input data and the reconstruction generated by their weights and the hidden units [42]. Ye et al. [43] suggested a heterogeneous deep learning architecture that combines an autoencoder and a multilayer RBM with an associative memory layer. Within this semi-supervised learning approach, the features were learned in an unsupervised fashion while malware and benign files were classified by a supervised model. A modified RBM which generates new features based on probability was proposed by Benchea and Gavrilut [44]. These features are network, registry and file system activities. The generated features were used to train a one-sided perceptron (OSP) algorithm. Their experiment was performed using 1.2 million files with 31,507 files containing malware. With newly generated 300 features, this was failed to outperform OSP-MAP model on a detection task, however, it successfully reduced the false positive rate.

Deep belief network (DBN) is composed of stacked RBMs, each of which is trained on the previous one. It is a type of a probabilistic generative model with numerous hidden and visible layers. DeepSign [38] is a DBN-based malware detection model learned from an unlabelled dataset. DBN was used together with a deep stack of denoising autoencoders to build an invariant representation of malware behaviour. Hou et al. [39] compared the performance of four different shallow machine-learning algorithms (SVM, multi-layered perceptron, decision tree and naïve base) and a DBN-based detection model using unknown android-based malwares. DBN outperformed all other shallow models. Ding et al. [40] developed a DBN-based malware detection algorithm and compared its performance with three machine-learning algorithms namely, SVM, decision tree and k-nearest neighbour. Their experimental results confirmed that the performance of their DBN-based model outperforms when more unlabelled data is included. Chen et al. [76] proposed a malware detection method for mobile edge computing (MEC). This method used a deep belief network, actively extracted attack features on MEC devices on the Android system. Comparing their model to four existing machine-learning-based detection techniques, the active feature learning of the model offered an advantage in the detection accuracy improvement.

2) SUPERVISED LEARNING

As supervised learning employs labelled data to learn a model, it generally gives lower false alarm rate than unsupervised learning. As a signature-based approach, supervised learning performs poorly in classifying novel or unseen malwares or variants [45].

Convolutional neural network (CNN) has widely been used for image classification tasks. To use a CNN for malware classification, features need to be transformed into images. Kalash et al. [46] proposed a CNN-based malware detection method. A decimal vector representation was generated by translating the binary values that were split into 8-bit sequences to decimal values. It was then reshaped into a 2D matrix to be displayed as a grayscale image. Their experiment demonstrated that the malwares from the same family could have the same visual representation. With the dataset from Microsoft's Malware Classification Challenge, they achieved a classification accuracy of 99.97%.

Abdelsalam et al. [47] compared the performance of a 2D CNN with a 3D CNN and their 3D CNN model outperformed 2D CNN model. Jeon et al. [48] introduced a new IoT malware detection technique based on 3D CNN. Features of memory data, integrated feature data (network, system call, process, and VFS) and behaviour frequency table were transformed into red, green, and blue channels and combined into a single RGB image. Their 512x512 vector size model achieved 99.28% accuracy. Yet, their dataset included only 1,401 samples that have not been cross validated.

Recurrent neural network is designed to learn a model from sequential or time series data by allowing previous outputs to be used as input while having hidden states [35]. Sun and Qian [49] studied to see if amount of labelled data could influence the performance of deep learning algorithms such as CNN and RNN. Here, RNN showed much improved performance in malware detection. Rhode et al. [50] compared the performance of RNN-based malware detection model with traditional machine-learning algorithms. Their ensemble of RNNs successfully identified whether an executable is malicious or benign within the first 5 seconds of running with 94% accuracy. Wang and Yiu [51] developed RNN-based autoencoders to address the interpretability issue. Multi-task learning was used to categorise malware as well as to develop file access patterns (FAPs) for API call sequences.

Vinayakumar et al. [52] used static and dynamic analysis to develop a long short-term memory (LSTM)-based android malware detection method. Their experiment showed that their stacked LSTM layer with 32 memory blocks which has a high number of epochs improved its performance. Fang et al. [53] used a LSTM model for detecting malicious JavaScript and its performance was compared with those of LSTM, naive bayes, SVM, and random forest models using bytecode sequences. LSTM showed the highest accuracy rate of 99.53%. An effective malware detection framework for Android devices on 5G networks is shown in the study of [82], which uses CNN and LSTM along with the opcodes feature called DLAMD for both rapid detection and the deep detection phase of malware detection.

Semi-supervised learning has also been popular as it may be able to tackle the limitations of supervised and unsupervised methods. Santos et al. [75] recommended using partially labelled data to limit the utilisation of labelled input data. They empirically showed that as the number of labelled data grows, the detection performance improves. Gamage and Samarabandu [54] compared the performance of two different semi-supervised learning algorithms (*i.e.*, autoencoder+multi-layered perceptron and DBN+multi-layered perceptron) and observed that semi-supervised learning does not improve the performance over the supervised models.

Relating to the fact that cyber threats greatly vary depending on the type of devices, operating systems, and networks used. A comparison table (Table 2) considering different API, datasets, analysis, and approaches is provided.

III. CONTINUOUS LEARNING

As discussed, deep learning allows the learning of complex and useful features in an automated fashion due to its extended capacity to mimic the human brain. However, it is evident that it could be assessed solely from the outside meaning that it is very difficult to understand how it makes decision [55]. With transfer learning, however, the learning process and performance could be improved when there is isolated learning paradigm by utilising related data. Li et al. [56] proposed a new concept that employs adaptive regularisation along with transfer learning that could transfer the learned model from its training to testing domain to identify unknown malicious samples. Although they first used this approach, their experiment was limited to one source domain to one target domain.

Lifelong machine learning (LML) gains attention in various areas these days due to its capability to transfers the knowledge of pre-learned models from different sources to multiple targets. One of the key requirements in data-driven malware detection is that the model should be adaptable to new malware variants and families as they change over time. In this sense, retaining and accumulating the knowledge learned in the past via LML's memory concept is vital. Fig. 1. depicts LML's key concepts. As indicated, the learned data is accumulated and used seamlessly in future learning.

Since Thrun and Mitchell's [57] first introduction of LML, the concepts used in LSM have been adopted in many different fields. Kozik et al. [58] proposed an LML-based intrusion detection system named B-ELLA which extends the ELLA [59] framework to overcome the problem of its imbalanced network data. Du et al. [60] developed a new LML-based anomaly detection method that uses unlearning to update their model. Their experimental results demonstrated the effectiveness of their method by reducing false positive and false negative rates significantly. Hong et al. [61] proved the usefulness of LML when there is insufficient labelled data.

Detecting malware with LML brings certain challenges when learning from i) unseen/unknown, and ii) imbalance

TABLE 2. Machine-Learning-Based Malware Detection

Author	Algorithm	Dataset	Feature	Environment
Huda *et al.* (2022) [92]	DBN	SCADA	Network traffic	IoT based SCADA
Patel *et al.* (2022) [89]	CNN	Malimg	Malware binaries	Autonomous vehicles
Ce *et al.* (2022) [99]	CNN+LSTM	VirusShare	API sequences	Windows
Yadav *et al.* (2022) [97]	CNN	R2-D2 + other sources	Bytecode	Android
Mohan *et al.* (2022) [98]	Adaboost	Software Informer, MalwareBazaar	API calls	Windows
Ravi *et al.*(2021) [96]	CNN+LSTM	IoMT-Malware, MS Malware	Byte sequences	IoMT
Aslan *et al.* (2021) [95]	RF	Das Malwerk, MalwareBazaar, Malware DB, Malshare, Tekdefense, ViruSign, VirusShare, KernelMode	System calls with corresponding features	Cloud
Basnet *et al.* (2021) [90]	CNN	VirusTotal	Assemblies	SCADA EVCS
Jeon *et al.* (2020) [48]	CNN	IoT devices	Memory, network, system call, process, VFS features	IoT (Linux)
Abdullah *et al.* (2020) [91]	RF	VirusTotal	System Calls	Android
Chen *et al.* (2019) [76]	DBN	Transportation	Permissions, APIs, Dynamic behaviour	Mobile/edge
Niu *et al.* (2019) [87]	XGBoost	Apps	Opcode	X86 IoT Autonomous vehicles

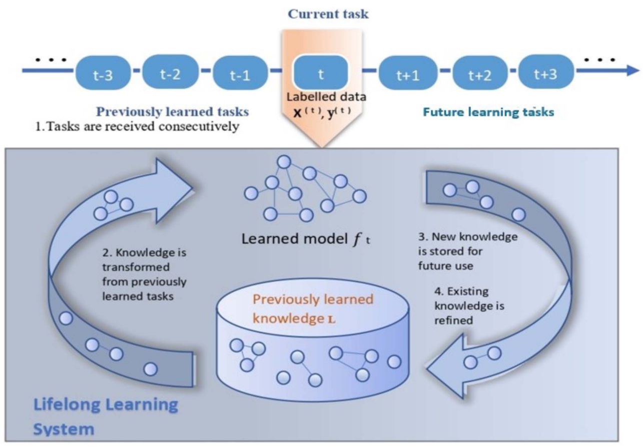
Current task
t-3
t-2
t-1
t
t+1
t+2
t+3
Labelled data
x(t), y(t)
Previously learned tasks
1.Tasks are received consecutively
Future learning tasks
Learned model f t
2. Knowledge is transformed from previously learned tasks
3. New knowledge is stored for future use
4. Existing knowledge is refined
Previously learned knowledge L
Lifelong Learning System

FIGURE 1. The concept of lifelong machine learning (LML). With utilising transfer learning and domain adaptation as a function, the learner has performed learning on a sequence of tasks, from t_1 to $t_{(n-1)}$. When faced with the nth task, it uses the relevant knowledge gained in the past n-1 tasks to help learning for the nth task.

and scarcity data. In a typical machine learning problem, models are learned purely from a given training data meaning that the models are unlikely to perform well when there are unseen/unknown samples in their testing data. During the LML phase, if the models are learned in an open environment where unseen/unknown samples appear unexpectedly then their learned models should be capable to classify such data accurately. As previously stated, LML could be seen as a continuous transfer learning procedure which accumulates the previously learned knowledge to assist or use new future learning. If a new LML-based malware detection mechanism learns in an open environment, it can detect novel or unknown malware variants accurately.

Obtaining a sufficient amount of labelled data for each class in malware detection is another challenge. Existing malware datasets have varying number of samples in each class. Machine learning in general performs well when it sees sufficient and balanced number of samples in each class. When limited labelled data, it would also be difficult to maintain a good performance of LML. The problem of few-shot recognition with unlabelled target data is largely unaddressed in the literature. Most recently, however, new concepts that could improve few-show learning with unlabelled dada have been proposed and showed promising results. STARTUP [62] is the first method that tackles this problem using self-training while it uses a fixed teacher pretrained on a labelled base dataset to create soft labels for the unlabelled target samples. Bateni et al [63] developed a transductive meta-learning method that uses unlabelled instances to improve the few-shot image classification performance.

IV. EXPLAINABILITY VIA VISUALISATION

Machine-learning-based methods seem to be most effective in malware detection, however, they are associated with a lack of transparency and explainability. When they can explain their predictions at an individual level accurately, it makes their models trustworthy. We believe the model trustworthiness could be achieved via several visualisation techniques reported in the literature.

Since numerous non-linear transformations performed during the network training, learning millions of parameters is too complex to make the model mathematically explainable. Feature and network visualisation techniques have been used to investigate the causes of exceptional performance. Saliency maps [64] and their modifications, Grad-CAM [65] and Smoothgrad [66] visualise where each input data arrives in the individual unit via gradient-based sensitivity maps. The outputs of Grad-CAM [65] determines 'where' and 'what' a learned model is looking at via identifying bias in the model's predictions. While feature visualisation can provide interpretable data at the unit level, it is difficult to interpret at the layer level. Kim et al. [67] successfully vectorised the training dataset prior to the learning process with convolution neural networks (CNNs) and identified common features of API call graphs with a high weight. Following the completion of the learning phase, they employed Grad-CAM that produces visual explanations from CNN-based models for malware detection. Caforio et al. [68] also employed the Grad-CAM approach to generate coarse localisation maps that emphasise the most critical regions of the traffic data representation that are likely to be explained with identical localisation maps for the same class for forecasting cyber-attacks. Iadarola et al.'s [69] Grad-CAM assisted the interpretation of the output malware classification by verifying the prediction reliability via exploiting activation maps – the samples belonging to the same family exhibit the same malicious payload (*i.e.*, the same malicious behaviour), that can be related to the areas interested by the activation maps as symptomatic of the malicious payload.

Understanding what a model has learned and how it makes its decision is essential for decoding the inner process of machine learning. As discussed, the most promising strategies are based on examining the activation process to identify relevant properties in a neuron level. It should, however, be noted examining activation process may restrict layer-level concepts. In terms of comprehending what a neural network learns at the layer level, concept-based approaches have shown favourable results [70].

Cao et al. [71] developed a neuron-level representation of learned features and a layer-level visualisation of feature maps in order to illustrate the misclassification of adversarial instances, a multi-level visualisation comprising of a network-level view of data flows. By identifying the paths for adversarial and benign data, they visualised the prediction process of a DNN successfully.

In cyber security, malware is generally identified in a monocular manner to express the attacking goals of their developers like Adware, Spyware, or Ransomware. Regardless of the composition of used paths or the attacking chain to achieve this goal, the categorisation process is not as simple as the definition of black and white colour. By investigating the development of a multi-label solution using machine learning, identifying malwares by their recipes or DNA is another under-researched area in the literature. Every malware type/class possesses certain characteristics, and if the regions of these characteristics could be segmented in a malware image, the resulting images of the samples from a particular class should reflect some common patterns. With the aforementioned development using Grad-CAM, we believe that common features could be located as in Fig. 2, that could be used to learn a model for prediction and for identifying their recipes or DNA.

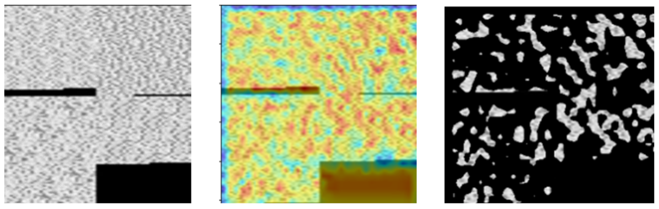
FIGURE 2. **The concept of extracted highlighted regions. Malware image (left), Grad-CAM visualisation (middle) indicating informative areas within the image and extracted highlighted regions for learning a model for identifying malwares by recipes (right).**

V. CONCLUDING REMARKS

In this survey, we provided an overview of the current state of the art of research in data-driven malware detection for 6G network. In particular, we discussed its theoretical and experimental aspects in the large-scale data-centric field, relating to: (1) continuous learning, including new concepts in multi-domain to multi-target learning and the challenges associated with unseen/unknown, imbalance data and data scarcity, and (2) new explainability via visualisation concepts with a multi-labelling approach which allows identifying malware by their recipes while improving the interpretability of its decision process. It is also envisaged that such data-modelling revolution in malware detection can be readily extended to various areas in cyber security. These newly designed continuous, explainable, and visualisible machine-learning-based approaches will not only be able to cope with the emerging issues associated with data-centric paradigm, but also provide a means in maximising its return for the various data modelling areas in which transparency in learning is required.

ACKNOWLEDGMENT

The authors are grateful to Dr Hussam Al-Hammadi at the Centre for Cyber-Physical Systems (C2PS) for his invaluable discussions and feedback, and special thanks to the EBTIC, British Telecom's (BT) security team in London and Abu Dhabi for their constructive criticism on this work.

REFERENCES

[1] Financial Crimes Enforcement Network (U.S.), "Ransomware trends in bank secrecy act data between January 2021 and June 2021," FinCEN advisory, Oct. 2021, [Online]. Available: https://www.fincen.gov/sites/default/files/2021-10/Financial%20Trend%20Analysis_Ransomware%20508%20FINAL.pdf
[2] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *J. Netw. Comput. Appl.*, vol. 153, 2020, Art. no. 102526.
[3] M. Macas, C. Wu, and W. Fuertes, "A survey on deep learning for cybersecurity: Progress, challenges, and opportunities," *Comput. Netw.*, vol. 212, Jul. 2022, Art. no. 109032, doi: 10.1016/j.comnet.2022.109032.
[4] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
[5] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Secur.*, 2011, pp. 1–7.

[6] VirusTotal, Accessed: Sep. 9, 2022, [Online]. Available: https://www.virustotal.com/gui/home/upload
[7] VirusShare.com, Accessed: Sep. 9, 2022, [Online]. Available: https://virusshare.com/
[8] MalwareBazaar, Accessed: Sep. 9, 2022, [Online]. Available: https://bazaar.abuse.ch/
[9] H. S. Anderson and P. Roth, "Ember: An open dataset for training static pe malware machine learning models," 2018, *arXiv:1804.04637*.
[10] "VX-heaven," Accessed: Sep. 9, 2022, [Online]. Available: https://academictorrents.com/details/34ebe49a48aa532deb9c0dd08a08a017aa04d810
[11] "The drebin dataset," Accessed: Sep. 9, 2022, [Online]. Available: https:www.sec.cs.tu-bs.de/~danarp/drebin/
[12] "The Microsoft malware classification challenge," Accessed: Sep. 9, 2022, [Online]. Available: https://www.kaggle.com/c/malware-classification/overview
[13] "Comodo cloud security center," Accessed: Sep. 9, 2022, [Online]. Available: https://www.comodo.cm/
[14] R. Sihwail, K. B. Omar, and K. A. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 8, no. 4–2, pp. 1662–1671, 2018, doi: 10.18517/ijaseit.8.4-2.6827.
[15] "VirusTotal API," Accessed: Sep. 9, 2022, [Online]. Available: https://developers.virustotal.com/reference/overview
[16] M. Rhode, P. Burnap, and K. Jones, "Early stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, 2018.
[17] R. B. Hadiprakoso, H. Kabetta, and I. K. S. Buana, "Hybrid-based malware analysis for effective and efficiency android malware detection," in *Proc. Int. Conf. Inform. Multimedia, Cyber Inf. Syst.*, 2020, pp. 8–12, doi: 10.1109/ICIMCIS51567.2020.9354315.
[18] M. Choudhary and B. Kishore, "HAAMD: Hybrid analysis for android malware detection," in *Proc. Int. Conf. Comput. Commun. Inform.*, 2018, pp. 1–4, doi: 10.1109/ICCCI.2018.8441295.
[19] P. Shijo and A. Salim, "Integrated static and dynamic analysis for malware detection," *Procedia Comput. Sci.*, vol. 46, pp. 804–811, 2015.
[20] H. E. Merabet and A. Hajraoui, "A survey of malware detection techniques based on machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 1, pp. 366–373, 2019, doi: 10.14569/IJACSA.2019.0100148.
[21] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *J. Syst. Architecture*, vol. 112, 2021, Art. no. 101861, doi: 10.1016/j.sysarc.2020.101861.
[22] E. Raff and C. Nicholas, "A survey of machine learning methods and challenges for windows malware classification," 2020, *arXiv:2006.09271*.
[23] T. -Y. Wang, C. -H. Wu, and C. -C. Hsieh, "Detecting unknown malicious executables using portable executable headers," in *Proc. 5th Int. Joint Conf. INC, IMS IDC*, 2009, pp. 278–284, doi: 10.1109/NCM.2009.385.
[24] F. Manavi and A. Hamzeh, "A new method for ransomware detection based on PE header using convolutional neural networks," in *Proc. 17th Int. ISC Conf. Inf. Secur. Cryptology*, 2020, pp. 82–87, doi: 10.1109/ISCISC51277.2020.9261903.
[25] N. Balram, G. Hsieh, and C. McFall, "Static malware analysis using machine learning algorithms on APT1 dataset with string and PE header features," in *Proc. Int. Conf. Comput. Sci. Comput. Intell.*, 2019, pp. 90–95, doi: 10.1109/CSCI49370.2019.00022.
[26] R. Zheng, Q. Wang, Z. Lin, Z. Jiang, J. Fu, and G. Peng, " Cryptocurrency malware detection in real-world environment: Based on multi-results stacking learning," *Appl. Soft Comput.*, vol. 124, 2022, Art. no. 109044, doi: 10.1016/j.asoc.2022.109044.
[27] Y. Ding, X. Zhang, and J. Hu, "Android malware detection method based on bytecode image," *J. Ambient Intell. Hum. Comput.*, pp. 1–10, 2020, doi: 10.1007/s12652-020-02196-4.
[28] Z. Salehi, A. Sami, and M. Ghiasi, "Using feature generation from api calls for malware detection," *Comput. Fraud Secur.*, vol. 2014, no. 9, pp. 9–18, 2014.
[29] M. Nar, N. Carkaci, A. Kakisim, and I. Sogukpinar, "Analysis and evaluation of dynamic feature-based malware detection," in *Proc. Int. Conf. Secur. Inf. Technol. Commun.*, Berlin, Germany: Springer, 2018, pp. 247–258.
[30] A. Mohaisen, O. Alraw, and M. Mohaisen, "Amal: High-fidelity, behavior based automated malware analysis and classification," *Comput. Secur.*, vol. 52, pp. 251–266, 2015.
[31] W. Han, J. Xue, Y. Wang, Z. Liu, and Z. Kong, "MalInsight: A systematic profiling based malware detection framework," *J. Netw. Comput. Appl.*, vol. 125, pp. 236–250, 2019.
[32] F. H. Hsu, C. W. Ou, Y. L. Hwang, Y. C. Chang, and P. C. Lin, "Detecting web-based botnets using bot communication traffic features," *Secur. Commun. Netw.*, early access, 2017, doi: 10.1155/2017/5960307.
[33] M. Eskandari, Z. Khorshidpour, and S. Hashemi, "Hdm-analyser: A hybrid analysis approach based on data mining techniques for malware detection," *J. Comput. Virology Hacking Techn.*, vol. 9, pp. 77–93, 2013.
[34] X. Ma, Q. Biao, W. Yang, and J. Jiang, "Using multi-features to reduce false positive in malware classification," in *Proc. IEEE Inf. Technol., Netw., Electron. Automat. Control Conf.*, 2016, pp. 361–365, doi: 10.1109/ITNEC.2016.7560382.
[35] D. Wei, Y. Wu, and J. Feng, "Network attacks detection methods based on deep learning techniques: A survey," *Secur. Commun. Netw.*, vol. 2020, pp. 1–17, 2020.
[36] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 3854–3861, doi: 10.1109/IJCNN.2017.7966342.
[37] A. Naway and Y. Li, "Android malware detection using autoencoder," 2019, *arXiv: 1901.07315*.
[38] O. David and N. S. Netanyahu, "DeepSign: Deep learning for automatic malware signature generation and classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–8.
[39] S. Hou, A. Saas, Y. Ye, and L. Chen, "Droiddelver: An android malware detection system using deep belief network based on api call blocks," in *Proc. Int. Conf. Web-Age Inf. Manage.*, Berlin, Germany: Springer-Verlag, vol. 9998, 2016, pp. 54–66.
[40] Y. Ding, S. Chen, and J. Xu, "Application of deep belief networks for opcode-based malware detection," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 3901–3908, doi: 10.1109/IJCNN.2016.7727705.
[41] S. Mahdavifar and A. Ghorbani, "Application of deep learning to cybersecurity: A survey," *Neurocomputing*, vol. 347, pp. 149–176, 2019.
[42] S. K. Kim, P. L. McMahon, and K. Olukotun, "A large-scale architecture for restricted boltzmann machines," in *Proc. 18th IEEE Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, 2010, pp. 201–208, doi: 10.1109/FCCM.2010.38.
[43] L. Chen, S. Hou, W. Hardy, Y. Ye, and X. Li, "Deepam: A heterogeneous deep learning framework for intelligent malware detection," *Knowl. Inf. Syst.*, vol. 54, no. 2, pp. 265–285, Feb. 2018, doi: 10.1007/s10115-017-1058-9.
[44] R. Benchea and D. T. Gavrilut, "Combining restricted boltzmann machine and one side perceptron for malware detection," in *Proc. Int. Conf. Conceptual Struct.*, 2014, pp. 93–103.
[45] P. M. Comar, L. Liu, S. Saha, P. -N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *Proc. IEEE INFOCOM*, 2013, pp. 2022–2030, doi: 10.1109/INFCOM.2013.6567003.
[46] 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.*, 2018, pp. 1–5, doi: 10.1109/NTMS.2018.8328749.
[47] R. Krishnan, M. Abdelsalam, Y. Huang, and R. Sandhu, "Malware detection in cloud infrastructures using convolutional neural networks," in *Proc. IEEE 11th Int. Conf. Cloud Comput.*, 2018, pp. 162–169.
[48] J. Jeon, J. H. Park, and Y. -S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, 2020, doi: 10.1109/ACCESS.2020.2995887.
[49] G. Sun and Q. Qian, "Deep learning and visualization for identifying malware families," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 283–295, Jan./Feb. 2021, doi: 10.1109/TDSC.2018.2884928.
[50] M. Rhode, P. Burnap, and K. Jones, "Early stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, 2018.
[51] X. Wang and S. Yiu, "A multi-task learning model for malware classification with useful file access pattern from api call sequence," 2016, *arXiv: 1610.05945*.

[52] R. Vinayakumar and K. P. Soman, P. Poornachandran, and S. S. Kumar, "Detecting android malware using long short-term memory (LSTM)," *J. Intell. Fuzzy Syst.*, vol. 34, pp. 1277–1288, 2018.

[53] Y. Fang, C. Huang, L. Liu, and M. Xue, "Research on malicious JavaScript detection technology based on LSTM," *IEEE Access*, vol. 6, pp. 59118–59125, 2018, doi: 10.1109/ACCESS.2018.2874098.

[54] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *J. Netw. Comput. Appl.*, vol. 169, 2020, Art. no. 102767.

[55] X. Hong, P. Wong, D. Liu, S.-D. Guan, K. L. Man, and X. Huang, "Lifelong machine learning: Outlook and direction," in *Proc. 2nd Int. Conf. Big Data Res.*, 2018, pp. 76–79.

[56] H. Li, Z. Chen, R. Spolaor, Q. Yan, C. Zhao, and B. Yang, "DART: Detecting unseen malware variants using adaptation regularization transfer learning," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6, doi: 10.1109/ICC.2019.8761598.

[57] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robot. Auton. Syst.*, vol. 15, no. 1-2, pp. 25–46, Jul. 1995, doi: 10.1016/0921-8890(95)00004-Y.

[58] M. Choraś, R. Kozik, and J. Keller, "Balanced efficient lifelong learning (B-ELLA) for cyber attack detection," *J. Univ. Comput. Sci.*, vol. 25, pp. 2–15, 2019.

[59] P. Ruvolo and E. Eaton, "Ella: An efficient lifelong learning algorithm," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 507–515.

[60] Z. Chen, C. Liu, R. Oak, M. Du, and S. Dawn, "Lifelong anomaly detection through unlearning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, pp. 1283–1297, 2019, doi: 10.1145/3319535.3363226.

[61] X. Hong, S.-U. Guan, K. L. Man, and P. W. Wong, "Lifelong machine learning architecture for classification," *Symmetry*, vol. 12, no. 5, 2020, Art. no. 852.

[62] A. Islam, C. F. Chen, R. Panda, L. Karlinsky, R. Feris, and R. J. Radke, "Dynamic distillation network for cross-domain few-shot recognition with unlabeled data," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 3584–3595, Dec. 2021.

[63] P. Bateni, J. Barber, J. -W. van de Meent, and F. Wood, "Enhancing few-shot image classification with unlabelled examples," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2022, pp. 1597–1606, doi: 10.1109/WACV51458.2022.00166.

[64] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*.

[65] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: 10.1007/s11263-019-01228-7.

[66] D. Smilkov, N. Thorat, B. Kim, F. Viegas, and M. Wattenberg, "Smoothgrad: Removing noise by adding noise," 2017, *arXiv:1706.03825*.

[67] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "Mapas: A practical deep learning-based android malware detection system," *Int. J. Inf. Secur.*, vol. 21, pp. 725–738, 2022, doi: 10.1007/s10207-022-00579-6.

[68] F. P. Caforio, G. Andresini, G. Vessio, A. Appice, and D. Malerba, "Leveraging grad-CAM to improve the accuracy of network intrusion detection systems," in *Proc. Int. Conf. Discov. Sci.*, Berlin, Germany: Springer-Verlag, vol. 38, 2021, pp. 385–400, doi: 10.1007/978-3-030-88942-5_30.

[69] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Evaluating deep learning classification reliability in android malware family detection," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops*, 2020, pp. 255–260, doi: 10.1109/ISSREW51248.2020.00082.

[70] A. Bäuerle, D. Jönsson, and T. Ropinski, "Neural activation patterns (NAPs): Visual explainability of learned concepts," 2022, *arXiv:2206.10611*.

[71] K. Cao, M. Liu, H. Su, J. Wu, J. Zhu, and S. Liu, "Analyzing the noise robustness of deep neural networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 7, pp. 3289–3304, Jul. 2021, doi: 10.1109/TVCG.2020.2969185.

[72] V. -L. Nguyen, P. -C. Lin, B. -C. Cheng, R. -H. Hwang, and Y. -D. Lin, "Security and privacy for 6G: A survey on prospective technologies and challenges," *IEEE Commun. Surv. Tut.*, vol. 23, no. 4, pp. 2384–2428, Oct.–Dec. 2021, doi: 10.1109/COMST.2021.3108618.

[73] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, and W. Zhou, "Security and privacy in 6G networks: New areas and new challenges," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 281–291, 2020.

[74] S. Shen, C. Yu, K. Zhang, J. Ni, and S. Ci, "Adaptive and dynamic security in AI-empowered 6G: From an energy efficiency perspective," *IEEE Commun. Standards Mag.*, vol. 5, no. 3, pp. 80–88, Sep. 2021, doi: 10.1109/MCOMSTD.101.2000090.

[75] A. Ankita and S. Rani, "Machine learning and deep learning for malware and ransomware attacks in 6G network," in *Proc. 4th Int. Conf. Comput. Intell. Commun. Technol.*, 2021, pp. 39–44, doi: 10.1109/CCICT53244.2021.00019.

[76] Y. Chen, Y. Zhang, S. Maharjan, M. Alam, and T. Wu, "Deep learning for secure mobile edge computing in cyber-physical transportation systems," *IEEE Netw.*, vol. 33, no. 4, pp. 36–41, Jul./Aug. 2019, doi: 10.1109/MNET.2019.1800458.

[77] M. S. Alam and S. T. Vuong, "Random forest classification for detecting android malware," in *Proc. IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Social Comput.*, 2013, pp. 663–669, doi: 10.1109/GreenCom-iThings-CPSCom.2013.122.

[78] I. Firdausi, C. lim, A. Erwin, and A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection," in *Proc. 2nd Int. Conf. Adv. Comput., Control, Telecommun. Technol.*, 2010, pp. 201–203, doi: 10.1109/ACT.2010.33.

[79] F. Shang, Y. Li, X. Deng, and D. He, "Android malware detection method based on naive bayes and permission correlation algorithm," *Cluster Comput.*, vol. 21, no. 1, pp. 955–966, 2018.

[80] Z. Markel and M. Bilzor, "Building a machine learning classifier for malware detection," in *Proc. 2nd Workshop Anti-Malware Testing Res.*, 2014, pp. 1–4, doi: 10.1109/WATeR.2014.7015757.

[81] A. N. Özalp, Z. Albayrak, M. Çakmak, and E. Özdoğan, "Layer-based examination of cyber-attacks in IoT," in *Proc. 2022 Int. Congr. Hum.-Comput. Interact., Optim. Robot. Appl.*, 2022, pp. 1–10, doi: 10.1109/HORA55278.2022.9800047.

[82] N. Lua, D. Lib, W. Shib, P. Vijayakumarc, F. Picciallid, and V. Change, "An efficient combined deep neural network based malware detection framework in 5G environment," *Comput. Netw.*, vol. 189, Apr. 2021, Art. no. 107932, doi: 10.1016/j.comnet.2021.107932.

[83] H. Haddadpajouh, A. Mohtadi, A. Dehghantanaha, H. Karimipour, X. Lin, and K. -K. R. Choo, "A multikernel and metaheuristic feature selection approach for IoT malware threat hunting in the edge layer," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4540–4547, Mar. 2021, doi: 10.1109/JIOT.2020.3026660.

[84] M. A. Rahman and M. S. Hossain, "A deep learning assisted software defined security architecture for 6G wireless networks: IIoT perspective," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 52–59, Apr. 2022, doi: 10.1109/MWC.006.2100438.

[85] H. Rafiq, N. Aslam, U. Ahmed, and J. C. Lin, "Mitigating malicious adversaries evasion attacks in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, early access, Jul. 07, 2022, doi: 10.1109/TII.2022.3189046.

[86] I. Ahmed, M. Anisetti, A. Ahmad, and G. Jeon, "A multi-layer deep learning approach for malware classification in 5G-enabled IIoT," *IEEE Trans. Ind. Inform.*, early access, Sep. 09, 2022, doi: 10.1109/TII.2022.3205366.

[87] W. Niu, X. Zhang, X. Du, T. Hu, X. Xie, and N. Guizani, "Detecting malware on X86-based IoT devices in autonomous driving," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 80–87, Aug. 2019, doi: 10.1109/MWC.2019.1800505.

[88] S. Huda et al., "Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data," *Inf. Sci.*, vol. 379, pp. 211–228, Feb. 2017.

[89] D. Patel et al., "Deep learning and blockchain-based framework to detect malware in autonomous vehicles," in *Proc. Int. Wireless Commun. Mobile Comput.*, 2022, pp. 278–283, doi: 10.1109/IWCMC55113.2022.9824186.

[90] M. Basnet, S. Poudyal, M. H. Ali, and D. Dasgupta, "Ransomware detection using deep learning in the SCADA system of electric vehicle charging station," in *Proc. IEEE PES Innov. Smart Grid Technol. Conf. - Latin Amer.*, 2021, pp. 1–5, doi: 10.1109/ISGTLatinAmerica52371.2021.9543031.

[91] Z. Abdullah, F. W. Muhadi, M. M. Saudi, I. R. A. Hamid, and C. F. M. Foozy, "Android ransomware detection based on dynamic obtained features," in *Proc. Int. Conf. Soft Comput. Data Mining*, 2020, pp. 121–129.

[92] S. Huda, J. Yearwood, M. M. Hassan, and A. Almogren, "Securing the operations in SCADA-IoT platform based industrial control system using ensemble of deep belief networks," *Appl. Soft Comput.*, vol. 71, pp. 66–77, 2018.

[93] M. R. Watson, N. -u. -h. Shirazi, A. K. Marnerides, A. Mauthe, and D. Hutchison, "Malware detection in cloud computing infrastructures," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 2, pp. 192–205, Mar./Apr. 2016, doi: 10.1109/TDSC.2015.2457918.

[94] M. Subedar, N. Ahuja, R. Krishnan, I. J. Ndiour, and O. Tickoo, "Deep probabilistic models to detect data poisoning attacks," 2019, *arXiv:1912.01206*.

[95] Ö. Aslan, M. Ozkan-Okay, and D. Gupta, "Intelligent behavior-based malware detection system on cloud computing environment," *IEEE Access*, vol. 9, pp. 83252–83271, 2021, doi: 10.1109/ACCESS.2021.3087316.

[96] V. Ravi, T. D. Pham, and M. Alazab, "Attention-based multidimensional deep learning approach for cross-architecture IoMT malware detection and classification in healthcare cyber-physical systems," *IEEE Trans. Comput. Social Syst.*, early access, Aug. 19, 2022, 2022, doi: 10.1109/TCSS.2022.3198123.

[97] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "EfficientNet convolutional neural networks-based android malware detection," *Comput. Secur.*, vol. 115, Apr. 2022, Art. no. 102622.

[98] P. M. Anand, P. V. S. Charan, and S. K. Shukla, "A comprehensive API call analysis for detecting Windows-based ransomware," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience*, 2022, pp. 337–344, doi: 10.1109/CSR54599.2022.9850320.

[99] C. Li, Q. Lv, N. Li, Y. Wang, D. Sun, and Y. Qiao, "A novel deep framework for dynamic malware detection based on api sequence intrinsic features," *Comput. Secur.*, vol. 116, 2022, Art. no. 102686, doi: 10.1016/j.cose.2022.102686.

DILARA T. UYSAL is currently working toward the Ph.D. degree in data-driven malware detection with the Department of Computer Science and Information Systems, Birkbeck College, University of London, London, U.K. Her research focuses on the theory and methodology of machine learning for large-scale real-world cyber security problems. She has successfully applied machine learning approaches to a wide range of problem domains including static, dynamic, and hybrid-based malware detection.

PAUL D. YOO (Senior Member, IEEE) is currently with the Department of Computer Science and Information Systems, Birkbeck College, University of London, London, U.K. He held academic/research posts in Sydney, Cranfield, and the Korea Advanced Institute of Science and Technology. He is the Editor of the IEEE Sustainable Computing and ACM Computing Surveys and holds more than 100 prestigious journal and conference publications in highly regarded IEEE/ACM journals. He is affiliated with the University of Sydney, Camperdown, NSW, Australia, and KAIST as a Visiting Professor. His research addresses two broad topics: advanced data analytics and cyber physical systems security.

KAMAL TAHA (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Texas at Arlington, Arlington, TX, USA. From August 2008 to August 2010, he was as an Instructor of computer science with the University of Texas at Arlington. From 1996 to 2005, he was an Engineering Specialist for Seagate Technology, USA, Seagate is a leading computer disc drive manufacturer in the U.S. Since 2010, he has been an Associate Professor with the Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, UAE. He has more than 90 refereed publications that have appeared in prestigious top ranked journals, conference proceedings, and book chapters. More than 20 of his publications have appeared in IEEE Transactions journals. His research interests span bioinformatics, information forensics and security, information retrieval, data mining, databases, and defect characterization of semiconductor wafers, with an emphasis on making data retrieval and exploration in emerging applications more effective, efficient, and robust. He is a Member of the Program Committee, Editorial Board, and Review panel for a number of international conferences and journals, some of which are IEEE and ACM journals.