ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
Received: 9 October 2025; Accepted: 7 January 2026

https://doi.org/10.17559/TV-20251009003047
Subject review

An Overview of Convolutional Neural Network-Based Static Malware Analysis Techniques

Aleksa KOMOSAR, Milan GNJATOVIC*, Darko STEFANOVIC, Nemanja MACEK, Dusan SAVIC, Teodora VUCKOVIC

Abstract: This paper provides an overview of convolutional neural network-based static malware analysis techniques. Three research questions are considered: Which architectures based on or related to CNNs are used in static malware analysis? Which datasets are used to support research in this field, and what are the associated challenges? To what extent are the obtained models evaluated? Three scientific databases (Scopus, Web of Science, and MDPI) are searched, and the PRISMA framework is used to conduct and transparently present 70 papers selected according to dedicated inclusion and exclusion criteria. The overview recognizes the recent trend of conceptualizing malware as a sequential structure with both local and long-term dependencies, the need to reconsider the notion of dataset balance, and the need for consistent and transparent application of the F1-score.

Keywords: convolutional neural network; dataset imbalance; F1-score; static malware analysis

1 INTRODUCTION

The field of cybersecurity is an emerging domain that requires sustained attention, as the risks and number of cyberattacks are growing rapidly. For instance, Abdullah M. Alnajim et al. [1] highlighted different layers of industrial IoT systems regarding vulnerability, threats, and attacks. They concluded that customized, real-time security systems are needed for more effective defense. In the field of Android technology-based applications, there are concerns about the great potential for personal data leakage and fraud [2]. Furthermore, Ali Alqahtani et al. [3] focus on web-based malicious software (malware) as an essential part of cybersecurity.

One of the main approaches to defending against malicious code and malware relies on artificial intelligence (AI), more specifically in the field of deep learning (DL) and machine learning (ML) [1]. In this field, one can distinguish between static, dynamic, and hybrid code analyses led by different features. This paper focuses on the particular research task of automatic static malware analysis based on convolutional neural networks (CNNs) [4-6]. An executable can be represented as a sequence of bytes, i.e., a sequence of integer values belonging to the range [0, 255]. This sequence can be further transformed into a matrix and represented as a grayscale image. At the specification level, malware visualization and classification are based on the observation of significant visual similarities in image texture for malware instances belonging to the same family [7].

Recently, the expanding usage of CNN-based models for the task of malware detection and classification led to the establishment of a specific research line, which is also sometimes referred to as visualized analysis [8], in line with the fact that CNNs represent a subtype of neural network tailored to grid-like data (e.g., images, etc.). Although there are many overviews of research on DL-based malware analysis, none adequately addresses the specific question of CNN-based malware analysis. To meet this desideratum, we provide a transparent, systematic overview of recent research on static malware analysis based on CNNs. The overview covers the five years between 2019 and 2023. In total, three scientific databases (Scopus, Web of Science, and MDPI) are searched, and the PRISMA framework is used to conduct and transparently present 70 papers selected according to dedicated inclusion and exclusion criteria.

The following research questions guide the overview.

RQ1: Which architectures based on or related to CNNs are used in static malware analysis?

RQ2: Which datasets are used to support research in this field, and what are the related challenges?

RQ3: To what extent are the obtained models evaluated?

The paper is organized as follows. Section 2 reviews existing systematic overviews in the domain of static malware analysis based on deep learning and shows that there is a need for a specific literature review addressing recent research on the application of CNNs in this domain. Section 3 describes the literature retrieval and selection protocol. Section 4 reports and discusses the results of the systematic literature review. Finally, Section 5 concludes the paper.

2 RELATED WORK

To indicate the importance of the intended systematic literature overview, this section covers existing and relevant systematic literature reviews, overviews, and surveys in the domain of static malware analysis based on deep learning.

Pascal Maniriho et al. [9] distinguish between DL and ML techniques and between static, dynamic, and hybrid analysis/detection methods. For instance, they demonstrate that 37 papers in 2022 were based on DL, compared to 8 in 2021 and 2 in 2020, indicating the expansion of DL techniques in malware detection. Additionally, ML-based studies used static analyses in 25 studies, accounting for 40.98% of the total number of papers. On the other hand, DL-based static malware analysis studies account for 73.68% of the total, indicating the dominance of static analysis over dynamic and hybrid analyses. Janaka Senanayake et al. [10] highlight models based on static analysis, with obtained accuracies below and above 90%. Also, they illustrate a wide range of static analyses in 65% of the considered studies. Additionally, Adeel Ehsan et al. [11] focus on malware detection for Android app permissions. This paper evidences that 13 out of 16 papers were based on static code analysis. Also, they consider accuracy the only evaluation metric for the models in their

review. However, in imbalanced datasets, accuracy is not the most relevant evaluation metric; e.g., the macro-average F1-score is more relevant. This evaluation-related aspect is considered in this contribution.

Matthew G. Gaber et al. [12] provide a systematic review of artificial intelligence in general, making a distinction between ML and DL, as well as the static, dynamic, and hybrid approaches in malware detection. Furthermore, they show that high performance can be achieved by applying static features. Kennedy E. Ketebu et al. [13] focus on CNN and primarily consider accuracy as the only evaluation metric for studies in general. However, 15 papers in the review highlight the limitations of the underlying datasets. Rami Sihwail et al. [14] illustrate the application of API calls, control flow graphs (CFG), opcodes, and N-grams as features in malware analysis. Furthermore, there are five papers with different classifiers and accuracy. Rahman Ali et al. [15] perform a systematic literature review on deep learning methods in general, showing CNN as the most used architecture for malware detection. As a valuable metric, they propose accuracy, but the significance of the F1-score is not clearly stated. Mujaheed Abdullahi et al. [16] also make a distinction between ML and DL algorithms. Additionally, they cover a more comprehensive set of evaluation measures than [10-15, 17], including precision, recall, and F1-score.

Furthermore, they consider the application of CNN in the detection of denial of service (DoS) attacks, which are one of the most common cyberattacks. On the other hand, Kajal Jaisinghani and Santosh Singh [17] promote Windows, Android, and IoT separately, with specific models that are used, accuracy, and study limitations.

However, they do not clearly state which scientific databases were used, and their search query is rather broad. On the other hand, Ya Pan et al. [2] focus on static analysis to detect malware. However, they cover the scope of Android platforms, showing that permissions and API calls are the most used features. They consider different machine learning models, including CNN, multilayer perceptron (MLP), recurrent neural network (RNN), long short-term memory (LSTM), and deep belief network (DBN).

Search queries and inclusion and exclusion criteria applied in the considered review are, in general, not stated transparently. It can be observed that, despite the existence of related work, there is currently no specific literature review addressing the application of CNN in static malware analysis. This desideratum is addressed in this contribution.

3 MATERIALS AND METHODS

This section represents a detailed insight into the materials and methods used to conduct the reported systematic literature review. To conduct a precise and transparent overview, the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) framework [18] is applied. The search is focused on three scientific databases: Scopus, Web of Science, and MDPI.

3.1 Search Strategy

The adopted search strategy is based on three key phrases: "malware detection", "convolutional neural network", and "static code analysis". We intentionally omitted the rather general key phrases of "deep learning", to focus our search on the CNN-based methodology. The final search query combines the selected key phrases and their synonyms in a Boolean expression and is customized for the considered databases as follows:

Scopus: TITLE-ABS-KEY ("malware detection" OR "malware classification" OR "malicious code" OR "malware visualization" OR "malware analysis") AND TITLE-ABS-KEY ("convolutional neural network" OR "cnn") AND TITLE-ABS-KEY ("static code analysis" OR "static analysis" OR "static program analysis");

Web of Science: ((ALL = ("malware detection" OR "malware classification" OR "malicious code" OR "malware visualization" OR "malware analysis") AND ALL = ("convolutional neural network" OR "cnn") AND ALL = ("static code analysis" OR "static analysis" OR "static program analysis")));

MDPI: malware classification AND convolutional neural network AND static analysis.

Papers from 2019 to 2023 are observed. After reviewing Scopus and Web of Science, 25 duplicates were identified. Accordingly, a search was conducted in the MDPI database of papers to expand the number of papers.

3.2 Study Selection Criteria

The inclusion criteria are: IC1: The paper's objectives are focused on static code analysis. IC2: The applied methodology is based on or related to CNNs. On the other hand, exclusion criteria are: EC1: The paper is unrelated to the considered topic and research questions. EC2: The papers consider only dynamic malware analysis, not static. EC2: The papers consider only dynamic malware analysis, not static. EC3: The applied models are not based on or related to CNNs. EC4: The paper is written in languages other than English. EC5: The paper is a theoretical, systematic literature review, overview, or survey.

3.3 Data Extraction

In this study, data extraction relates to the following aspects of the papers: authors, title, year of publishing, link to the Scopus, Web of Science, or MDPI databases, abstract, author keywords, publisher, document type, summary of the paper, used dataset(s), applied architecture(s), type of classification task, evaluation process and results, and limitations.

3.4 PRISMA Workflow

A transparent representation of the paper selection process is shown in Fig. 1. In the initial identification phase, 161 papers were retrieved based on search queries. From this set, 34 duplicated papers were removed.

In the second phase, the remaining 127 papers were analyzed with respect to their titles, abstracts, and keyword lists and screening required detailed overviews. Based on this analysis, an additional 24 were removed due to the exclusion criteria, with the following distribution: one paper due to EC1, seven papers due to EC2, five papers due to EC3, one paper due to EC4, and ten papers due to EC5.

In the third phase, the remaining 103 papers were sought for retrieval, but 14 were not retrieved.

In the last phase, the remaining 89 papers underwent full-text analysis. Based on this analysis, the additional 19 papers were excluded due to the exclusion criteria, with the following distribution: three papers due to EC1, ten due to EC2, five due to EC3, and one due to EC5. Finally, the remaining 70 papers were included in this systematic literature review.

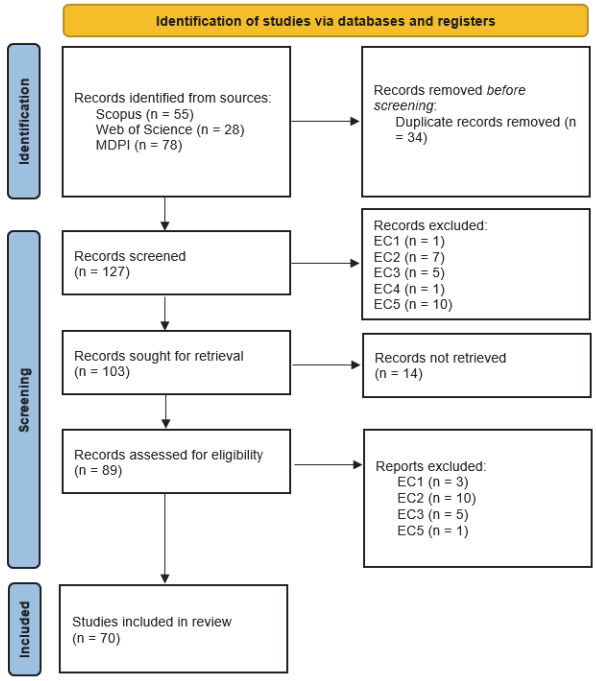

Figure 1 The PRISMA workflow

3.5 Threats to Validity

This systematic review is based on methodology and reporting using the PRISMA workflow. Each part of the systematic literature review is transparent and reflects the current state, as clarified by the adopted set of criteria. Fourteen papers were not retrieved, which could potentially contribute to the results and answers to the research questions. However, based on the analysis of these papers with respect to their titles, abstracts, and keyword lists, we believe that they would not make any substantial change to the reported results.

4 RESULTS AND DISCUSSION

The distribution of the selected papers by year of publication is the following: 10 in 2019, 13 in 2020, 14 in 2021, 13 in 2022, and 20 in 2023. Regarding paper type, 53 are research articles, and 17 are conference papers. The analysis of the selected papers with respect to the research questions stated in the Introduction is given in the following subsection.

4.1 RQ1: Which Architectures Based on or Related to CNNs are Used in Static Malware Analysis?

The diversity of architectures based on or related to CNNs is summarized in Tab. 1. All the architectures reported in the considered papers are grouped according to two criteria: the type of classification (i.e., multi-class vs binary) and the year of publication.

(i) Year 2019: Ten papers were published in 2019, and the dominantly applied architecture is a standard CNN. Yuxin Ding et al. [19] use a 2D-CNN for malware classification based on function call graphs and opcode sequences. They used two different CNN models, in which the output refers to the full connection layer. HiromuYakura et al. [20] add the attention mechanism to a 2D-CNN to capture important regions that have a higher priority in the classification process.

On the other hand, some authors use more specific variants of CNN. For instance, Di Xue et al. [21] use a CNN based on VGGNet-16, which consists of the input layer, 13 convolutional layers, five pooling layers, 3 fully connected layers, and one output layer. They set the last pooling layer to apply the spatial pyramid pooling instead of the max pooling. One of the main techniques to prevent overfitting is the application of a dropout regulation layer that randomly drops out a proportion of the neurons during training.

Additionally, Ajay Singh et al. [22] use ResNet-50, where every layer is responsible for the fine-tuning of the output from previous layers. The last architecture used for multi-classification in 2019 was proposed by Hyun-Kyo Lim et al. [23]. The model has two convolutional layers and one LSTM layer, where the lower layers focus on understanding small patterns and immediate relationships in the data. In comparison, the higher layers zoom out to grasp larger patterns and longer-term connections.

Table 1 Overview of used architectures

Year	Type of classification	Architecture(s)/Model(s)	Paper(s)
2019	Multi-class	M-CNN	[21]
		ResNet-50 + dense CNN	[22]
		CNN	[19]
		CNN + Attention Mechanism	[20]
		CNN+LSTM	[23]
	Binary	CNN	[24-27]
		Text-CNN	[28]
2020	Multi-class	CNN	[29-31]
		CNN + Attention Residual Module	[32]
		MalFCS	[33]
		AMalNet	[34]
		X-CNN	[35]
		GAN	[31]
		AlexNet and Inception-V3	[36]
		MCSP and MCSLT	[37]
		ResNet	[29]

Table 2 Overview of used architectures (continuation)

Year	Type of classification	Architecture(s)/Model(s)	Paper(s)
2020	Binary	CNN	[29, 38]
		CNN + LSTM + BiLSTM	[39]
		GAN	[40]
		CNN + SC-LSTM	[41]
2021	Multi-class	CNN	[42, 43]
		RMVC	[44]
		SACNN + VGG19	[45]
		ResNet + LSTM	[46]
		VGG16, AlexNet, DarkNet-53, DenseNet-201, Inception-V3, Places365-GoogleNet, ResNet-50, and MobileNet-V2	[47]
		AdvAndMal (CNN + CGAN)	[48]
		GCN - DGCNDroid	[49]
		FGNN	[50]
	Binary	CNN	[42]
		CNN-BiLSTM-LocAtt	[51]
		InceptionV3, ResNet50, ResNet101, DenseNet121, NASNet, and InveptionResNetV2	[52]
		CNN–BiLSTM–Attention	[53]
		1D-CNN, CNN-LSTM	[54]
		TCN, CNN, MobileNetV2	[55]
2022	Multi-class	CNN	[56]
		CNN + VGG16	[57]
		2HM-CNN	[58]
		CNN-LSTM	[59]
		FT-CFNN	[60]
		AIFS-IDL, GRU, TCN, CNN	[61]
		TCN-BiGRU	[62]
		ResNet18, MobileNetV2, and DenseNet161	[63]
	Binary	CNN (Image Load (IL), Single Load (SL), and Double Load (DL))	[64]
		TCN-BiGRU	[62]
		CNN	[65]
		LSTM	[66]
		MSFDroid	[67]
	N/A	CNN-LSTM	[68]
2023	Multi-class	ResNet1D	[69]
		CNN, CNN-LSTM	[3]
		ResNet50, MobilenetV1, and MobilenetV2	[70]
		VGG16, ResNet50, InceptionV3, VGG19, MobileNet, DenseNet169, Xception, DenseNet201, Inception ResNet V2, MobileNetV2, ResNet152V2, NasNetMobile, AlexNet, SqueezeNet, RegNetY320	[71]
		CNN, GAN	[72]
		XceptionNet, EfficientNetB0, ResNet50, VGG16, DenseNet169, and InceptionResNetV2	[73]
		DenseNet121, EfficientNetB0, VGG16, ResNet-101, MobileNetV1	[74]
		A neural network optimized with an improved version of HHO	[75]
		VGG-16 + CNN1 + CNN2	[76]
		Vision Transformer (ViT)	[77]
		GCN + CNN	[78]
		CNN, CNN-LSTM, and CNN BiLSTM	[79, 80]
	Binary	ResNet50, AlexNet, InceptionV3, ResNet101, GoogleNet, VGG16, DarkNet53, Xception, InceptionResNetV2, MobileNetV2, NasNetMobile, DarkNet19, ResNet18, DenseNet201, NasNetLarge, Places365-GoogleNet, ShuffleNet, SqueezeNet, VGG19	[8]
		CNN-LSTM	[81]
		GCN	[82]
		GhostNet	[83]
		CNN	[84]
	Multiclass and clustering	LSTM, AE, LSTM-CNN, and CNN NLP	[85]
	N/A	CNN	[86]

The remaining papers from 2019 are focused on binary classification. The base architecture used is again CNN [24-27], while Yongjun Lee et al. [28] add one convolutional layer for word vectors derived from an unsupervised neural language model.

(ii) Year 2020: In 2020, the applied CNNs are extended by additional layers and mechanisms. Asim Darwaish and Farid Naït-Abdesselam [29] compare CNN and ResNet with no significant performance improvement indicators, while [30] illustrate recurrent plots for visualizing recurrence in addition to CNN. Sejun Jang et al. [31] propose a framework that includes a GAN and a CNN. GAN is proposed in the training and classification phase as the trainer for local images. Additionally, the GAN executor is the next step for the trained GAN, which is followed by the CNN trainer and executor at the end. To resolve performance degradation in ResNeXt, Diangarti Bhalang Tariang et al. [32] integrate the attention mechanism into residual blocks. They show that this architecture may compensate for some resource limitations (e.g., computational). On the other hand, a framework called MalFC [33] includes an entropy graph and a feature

extractor based on a deep CNN with an additional SVM classifier.

Furthermore, to observe and learn more about malware's attributes, AMalNet [34] consists of a graph convolutional network for semantics, as well as an Independently Recurrent Neural Network to decode information from it. In [35], CNN is modified based on the construction of max-pooling layers in parallel for the X-CNN architecture. AlexNet and Inception-V3 are used for feature extraction [36]. Young-Man Kwon et al. [37] use two 2 × 2 × 32 convolutional filters before 2 × 2 max pooling, which is followed by one 2x2x32 convolutional filter and 2 × 2 max pooling. In the end, there are fully connected layers that show classification output.

For binary classification, CNNs are applied in [38-41]. Additionally, in [39] Lu Wang et al. also apply Long Short-Term Memory and Bidirectional Long Short-Term Memory, while on the other hand, Sunoh Choi et al. [41] add Skip-Connected to their LSTM-based model.

(iii) Year 2021: Keshav Thosar et al. [42] use 1D-CNN for binary as well as multi-classification. On the other hand, Chenyue Wang et al. [48] propose a model that uses a two-layer network for adversarial training. The first layer generates adversarial samples using a conditional generative adversarial network. The second layer of the network is responsible for malware classification based on RGB images generated from sequences of system calls. Additionally, for Android malware multi-classification based on API calls, a graph convolutional network is used [49]. Besides RNN, Guosong Sun and Quan Qian [44] use CNN and VGGNet. Mazhar Javed Awan et al. [45] use VGG19 and spatial convolutional attention. Additionally, the VGG variant - VGG16 was used in [47] beside AlexNet, DarkNet-53, DenseNet-201, Inception-V3, Places365-GoogleNet, ResNet-50, and MobileNet-V2. Another paper [50], based on the analysis of static API functions, applies a fuzzy graph convolutional network and provides insight into the complexity of algorithms that represent the characteristics of malware.

Furthermore, using variants of CNNs and combining them with other DL architectures is appropriate when addressing more demanding challenges. Chao Ding et al. [46] propose Res7LSTM, a model that is related to ResNet and LSTM. ResNet is used to decrease the problem of accuracy degradation as the depth of the network increases.

In the binary classification task, standard CNNs were dominantly applied in 2021. CNN combined with Bi-LSTM and local attention network makes the CNN-BiLSTM-LocAtt model applied in [51], where all of the networks have specific tasks. Bidirectional Long Short-Term Memory enables the model to capture the context in two directions, from past and future, for a better understanding of data, while Location-based attention mechanisms focus on the relevant part of the specific input representing static features. Additionally, Nan Zhang et al. [53] also provide a model similar to [51]. The main difference between these models is the range of action. Mathew Ashik et al. [54] provide 1D-CNN and LSTM among many other ML and DL-based models for classification, while Wenhui Zhang et al. [55] use CNN and MobileNetV2. In one of the approaches with the widest range of applied architectures, Seung-Pil W. Coleman et al. [52] use InceptionV3, ResNet50, ResNet101, DenseNet121, NASNet, and InceptionResNetV2 to test the performance of binary classification.

(iv) Year 2022: Lin Li et al. [56] use a CNN architecture, which consists of 5 layers, while the self-encoding architecture relies on two encoders, 16x16x128 and 32x32x256, respectively, including linear-relu and softmax. On the other hand, in [57] a CNN is complemented by VGG16 as the pre-trained weight of the visual geometry group, while Shakhnaz Amenova et al. [59] apply an LSTM besides a CNN to improve the prediction accuracy. As one of the trends in 2022, a more comprehensive set of architectures is used to compare and report the malware analysis results. For instance, GRU, TCN, and CNN are compared with the model AIFS-IDL proposed in [61]. Some authors use standard CNN variants. For instance, ResNet18, MobileNetV2, and DenseNet161 are used in [63]. While, on the other hand, some authors upgrade TCN[62] with the bidirectional gated recurrent units for multi-class and binary classification.

One of the novel models in 2022 is 2HM-CNN, which consists of 3 blocks, and every block includes two Conv2D layers, one max pooling layer, and a dropout function [58]. On the other hand, FT-CFNN[60] is based on two convolutional layers and a pooling layer between them. Additionally, a pooling layer was added before the feature fusion layer and fuzzy neural network.

For binary classification, CNN-based architectures are mainly applied, e.g., [65]. In addition, Islam Obaidat et al. [64] make a distinction between image load, single load, and double load. LSTM-related models are proposed in [66,68], but [68] does not specify the type of classification. Tao Peng et al. [67] propose MSFDroid for Android malware detection.

(v) Year 2023: In 2023, there are many variants and architectures used to build models that were the object of investigation in the context of performance, resources, and evaluation metrics. Ngo et al. [69] provide ResNet1D, which is a variant of 1D-CNN, with a high stride step (up to 8) within each ResNet1D block. That kind of block consists of two paths, one for learning and the other for residual shortcuts. Additionally, ResNet is mainly used in 2023 [70, 71, 73, 74] for multi-classification. In [71] besides ResNet, other models are also applied, including VGG16, InceptionV3, VGG19, MobileNet, DenseNet169, Xception, DenseNet201, Inception ResNet V2, MobileNetV2, ResNet152V2, NasNetMobile, AlexNet, SqueezeNet, and RegNetY320.

Another trend in 2023 is related to the application of LSTM. In [3, 79-81, 84-86], LSTM is integrated with CNNs. Additionally, some authors [72] use a 1D-CNN and GAN architecture, while others [78] focus on a 1D-CNN and GCN. As a novel architecture, [77] provides a Vision Transformer. Finally, for binary classifications, [8] proposes ResNet50, AlexNet, InceptionV3, ResNet101, GoogleNet, VGG16, DarkNet53, Xception, InceptionResNetV2, MobileNetV2, NasNetMobile, DarkNet19, ResNet18, DenseNet201, NasNetLarge, Places365-GoogleNet, ShuffleNet, SqueezeNet, and VGG19.

(vi) Conclusion: With respect to the methodological approach, the selected papers can be divided into two groups: 38 papers (i.e., 54.3 percent) report on approaches that are based on CNNs, while the remaining 32 papers

(i.e., 45.7 percent) propose approaches that integrate CNNs with other DL models. A diversity of models based on or related to CNNs shows a shift in the conceptualization of malware in the ML scientific community. In the beginning, CNN-based architectures were dominantly applied. This practice was in line with the dominant, although somewhat implicit, conceptualization of malware as a grid-like structure with local dependencies.

With time, this conceptualization has been extended, and malware was considered as a sequential structure with both local and long-term dependencies. This conceptual change is reflected in the integration of recurrent neural networks in CNN-based architectures and the ever-increasing depth of neural network models. However, although the current conceptualization is more comprehensive, the question of whether and to what extent it improves the classification performance is inconclusive. A part of the reasons is discussed in the following sections.

4.1 RQ2: Which Datasets are Used to Support Research in this Field, and what are the Related Challenges in Datasets?

A summarized overview of the datasets underlying the studies reported in the selected paper is provided in Tab. 2. For each dataset, the table provides the number of instances, the malign/benign ratio, the number of classes, and related papers. Most of them are available upon request, some of them are not publicly available (cf. [25, 27, 69]).

Table 2 Overview of datasets

Dataset	# Samples			Description	Related paper(s)
	Malware	Benign	Total		
Malimg [7]	9,339	N/A	9,339	25 malware classes, grayscale image representation	[22, 32, 33, 35, 36, 45, 47, 60, 70-74, 79, 80]
Microsoft Malware Classification Challenge (BIG 2015) [95]	21,741	N/A	21,741	9 malware classes, grayscale image representation (for each instance, both .byte and .asm files are provided)	[30-33, 37, 44, 56, 61, 62, 71, 74]
AndroZoo [88]	N/A	N/A	More than 15 mil	7 malware classes (dataset is constantly evolving)	[29, 34 ,48, 77, 78, 84]
Drebin [92]	5,560	N/A	5,560	179 malware classes	[19, 34, 48, 49, 52-54, 67, 75, 77, 78, 82, 84, 85]
CICMalDroid2020 [96]	N/A	N/A	More than 17,341	Multiple malware classes (incl. adware, banking malware, SMS malware, potentially harmful software)	[55, 59, 65, 67, 77, 78, 82]
CICAndMal2017 [97] and follow-up dataset 2019 [98]	N/A	N/A	More than 10,854	4 malware classes (adware, ransomware, scareware, and SMS malware)	[45, 54, 64, 66, 74, 80]
CICMalMem2022 [99]	29,298	29,298	58,596	15 malware classes	[66]
AMD [100]	24,553	N/A	24,553	17 malware classes	[34, 43, 52, 84]
ClaMP[101]	2,722	2,488	5,210	1 malware and 1 benign class	[3]
Benign & Malicious PE Files [102]	14,600	5,010	19,612	1 malware and 1 benign class	[3]
MalwareDataSet [103]	40,918	96,526	137,444	1 malware and 1 benign class	[3]
SEL [104]	500	9,500	10,000	1 ransomware and 1 benign class	[8]
EMBER 2018 [89]	400,000	400,000	More than 1 mil	1 malware and 1 benign class (in addition, there are 300,000 unlabeled instances)	[42]
Dumpware10 [105]	3,686	608	4,294	10 malware classes and 1 benign class	[57]
Android Botnet [91]	1,929	N/A	1,929	14 botnet classes	[82]
IoT-23 [106]	N/A	N/A	N/A	20 malware captures and 3 benign captures of IoT traffic	[83]
MaMaDroid [107]	35,500	8,500	44,000	N/A	[85]
MalGenome [93]	1,200	N/A	1,200	multiple malware classes	[85]
Andro-Dumpsys [90]	906	N/A	906	13 malware classes (binary files)	[76]
MaleVis [108]	N/A	N/A	More than 14,000	25 malware classes and one benign class (RGB images)	[63, 79]
Praguard Dataset [109]	N/A	N/A	10,479	Android-related instances	[34]

(i) Dataset representativeness. Under the notion of dataset representativeness, we refer to the extent to which instances comprised in a dataset manifest variability in malware and benign instances [87]. In the field of machine learning, dataset representativeness is a fundamental requirement for any claim on a generalization of the obtained results. It is important to note that we do not consider here representativeness in a general manner, but rather in a domain-specialized manner. Most of the considered datasets are domain-specific (e.g., Android-specific malware, etc.). However, even specialized datasets should manifest variability in domain-specific malware, and their representativeness may be evaluated by "the degree of closure" [87].

Although the representativeness of a dataset, even a specialized one, is an elusive ideal, it can be observed that the considered datasets differ to a significant extent with respect to their representativeness. The number of instances in some datasets is commendable, e.g., AndroZoo [88] contains more than 15 million instances divided into 7 malware classes, and EMBER 2018 [89]

contains more than one million instances divided into two classes (one malware and one benign).

On the other side of the scale, some datasets contain a relatively small number of instances distributed across several classes, e.g., Andro-Dumpsys [90] contains 906 instances divided into 13 classes, Android Botnet [91] contains 1,929 instances divided into 14 classes, Drebin [92] contains 5,560 instances divided into 179 classes, MalGenome [93] contains 1,200 instances divided into multiple classes. Such a dataset design does not allow conclusions about representativeness.

Some authors seem aware of the limited dataset representativeness. Thus, to increase the level of representativeness, they resort to several datasets. For example, the Microsoft Malware Classification Challenge dataset (BIG 2015) is combined with the Malimg dataset in [32, 33, 74]. In [73] the authors merge the Malimg and Malevis datasets. Authors in [46, 55, 67, 81] combine Android malware datasets. Also, the Drebin dataset is used in addition to the AndroZoo and AMD datasets in [34]. In [49], the Drebin dataset is used in combination with the Mobile Assistant[1] platform. Furthermore, [78, 84] combine the AndroZoo dataset with the Drebin, AMD, VirusShare, and CICMalDroid2020 datasets. Finally, Pratyush Panda et al. [79] use the Malimg and MaleVis datasets.

(ii) Dataset balance. In the ML-based malware analysis community, the dominant understanding of the notion of dataset balance refers to a dataset whose classes are represented with equal or similar numbers of instances [94]. The main concern related to imbalanced datasets is that they may introduce a bias toward a dominant class. In the context of this understanding, most of the observed datasets are imbalanced, and some authors try to address this problem. Generally, there are two ways to balance a dataset: by oversampling the underrepresented classes and undersampling the overrepresented classes.

(ii) Dataset balance. In the ML-based malware analysis community, the dominant understanding of the notion of dataset balance refers to a dataset whose classes are represented with equal or similar numbers of instances [94]. The main concern related to imbalanced datasets is that they may introduce a bias toward a dominant class. In the context of this understanding, most of the observed datasets are imbalanced, and some authors try to address this problem. Generally, there are two ways to balance a dataset: by oversampling the underrepresented classes and undersampling the overrepresented classes.

The oversampling techniques are relatively frequently applied in the reported studies, including augmentation techniques (e.g., based on the affine transformations) [36, 47, 71, 73, 79, 80], the SMOTE oversampling [65], function logic shuffling, junk code insertion, and function splitting [76]. However, it should be noted that the application of oversampling techniques for malware datasets is not necessarily methodologically justifiable, since the automatic generation of synthetic malware could negatively affect dataset representativeness (cf. [94]).

On the other hand, the undersampling techniques are applied less frequently in the considered paper set. In [72] a random selection of only 20 samples per class in the Malimg dataset is applied, resulting in just 500 instances. In [32] a more sophisticated weighted random sampling is applied, in which the weight of a class is initialized by the inverse of the number of its instances. In addition, some authors remove the duplicate instances to reduce the bias [22, 53, 75]. The presence of a clone problem (i.e., the presence of very similar malware instances) is considered only in [35, 80].

However, the question of why a dataset with equally represented classes should be considered representative is not considered (at least not in the manner of [94]). An interesting example is given in [44]. The authors intentionally decide not to balance a set of 3557 instances distributed over six classes, although they are aware of the class imbalance. We interpret their decision as being in line with an alternative understanding of the notion of balance, which originates from corpus linguistics but is significantly less frequently adopted in the ML-based malware analysis community. This understanding assumes that the number of instances across classes should be proportional to their frequencies in real life [87]. However, even [44] does not discuss the representativeness of the class proportion of the relatively small underlying dataset. A summary of the techniques applied to address the class imbalance problem is given in Tab. 3.

Finally, the considered datasets can be divided into two groups: static sample datasets and dynamic monitor datasets. Static sample datasets are datasets that are produced at one moment and do not undergo further resampling. Dynamic monitor datasets are datasets that are updated regularly ([22, 26, 51, 54, 58, 64, 71, 75, 84], cf. also studies [18, 21, 53, 71] which resorts to the VX Heaven website[2]). It may appear that balance is more important for static datasets (cf. [87]). However, researchers often process only selected parts of dynamic monitor datasets, and the instance selection reinforces the balance problem.

4.3 RQ3: To what extent are the obtained models evaluated?

The models proposed in the selected papers are generally reproducible. However, they are also complex, which makes their reproduction a time-demanding task and sometimes not practically feasible. Therefore, the reported evaluation results play an important role in any comparative analysis. To evaluate the models introduced in the selected papers, a set of evaluation measures has been applied (introduced in [6, 110, 111]). The application of different measures is summarized in Tab. 4.

Accuracy represents the most frequently applied measure (i.e., in 91.42% of papers). Tabs. 5, 6, and 7 provide information on the accuracy, precision, and recall values obtained on the most popular malware datasets: Malimg, Microsoft BIG 2015, and AndroZoo, respectively. In some cases, researchers provide additional evaluation measures. For instance, in [35], accuracy, recall, and precision are calculated for different malware families separately. The highest precision is obtained for the C2LOP.P family (i.e., 76%) and the highest recall for the C2LOP.gen!g family (i.e., 80%). Still, in more than half of the presented papers in Tabs. 5, 6, and 7, accuracy is the only provided evaluation measure.

[1]http://zhushou.360.cn/

[2]https://web.archive.org/web/20170611163424/http://vxheaven.org/

Table 3 Addressing the class imbalance problem

Paper	Dataset	Proposed technique to reduce imbalance	Accuracy
[56]	Microsoft Malware Classification Challenge (BIG 2015)	The dataset with the least number of instances is removed.	98,68%
[50]	N/A	N/A	96,7%
[3]	CLaMP, Benign and Malicious PE Files, MalwareDataSet	The datasets underwent data normalization, where the numerical values were transformed into a standardized format using Standard Deviation Normalization.	99,01%
[43]	AMD + APKure	N/A	94.13%
[44]	Microsoft Malware Classification Challenge (BIG 2015)	Mixing and randomly dividing datasets. Using different datasets for measurements.	99.5%
[30]	Microsoft Malware Classification Challenge (BIG 2015)	N/A	97.4%
[19]	DREBIN + Anzhi application store	N/A	97.5%
[76]	Andro-Dumpsys	Generating three additional synthetic malware variants, each of which was applied using different control flow obfuscation techniques.	AUC-ROC = 95.6%
[80]	Malimg	Randomly splitting the dataset, using the precision-recall curve, and model scaling.	N/A
[60]	Malimg	N/A	98.61%
[49]	DREBIN + Mobile Assistant platform	N/A	98.2%
[36]	Malimg	Image augmentation techniques.	95.2%
[28]	N/A	N/A	91.11%

Table 4 Overview of the applied measures

Measure	Percentage of papers
Accuracy	91.42%
Recall	70%
Precision	71,42%
True positive rate	10%
False positive rate	18,57%
False negative rate	7,14%
F1-score	75,71%
Area Under the ROC Curve	17,14%
Jaccard distance	1,42%
Cohen's Kappa	2,85%
Fowlkes-Mallows Index	1,42%
Matthews Correlation Coefficient	5,71%
Macro Recall	5,71%
Macro Precision	5,71%
Macro F1-score	5,71%

The combination of CNN and ARM architectures has yielded the highest accuracy results for the Malimg dataset, with a score of 99.96%. Furthermore, the MalFCS architecture has displayed an accuracy score of 100% on the Microsoft BIG 2015 dataset. The AmalNet architecture has achieved an accuracy of 99.69% on the AndroZoo dataset. In addition, it can be observed that the classification performance has been somewhat improved with respect to traditional non-connectionist image processing approaches. E.g., for the Malimg dataset:

A traditional, non-connectionist image processing approach based on Gabor filtering provides an accuracy of 97.18 percent. It has been observed that there is confusion between the closely similar malware families C2Lop and Swizzor. When these malware families are merged, the obtained accuracy is 99.2 percent [7].The accuracy obtained on the same dataset by applying models that are based on or related to CNNs is slightly improved and belongs to the range [96.54-99.97] (cf. Tab. 5). Interestingly, the problem related to the similarity between malware families C2Lop and Swizzor is still acknowledged [35, 80], but also addressed [47].

Table 5 Accuracy obtained on the Malimg dataset

Paper	Model	Accuracy	Precision	Recall
[70]	VMCTE (ResNet50, MobilenetV1, and MobilenetV2)	99.64%	99.66%	99.64%
[35]	X-CNN	96.54%	N/A	N/A
[60]	FT-CFNN	98.61%	N/A	N/A
[45]	SACNN + VGG16	97.62%, 97.42%	97.68% 97.11%	97.5% 96.95%
[47]	FT CNN	99.97%	99.04%	99.01%
[36]	AlexNet and Inception-V3	99.3%	N/A	N/A
[32]	CNN + ARM	99.96%	N/A	N/A
[33]	MalFCS	99.72%	N/A	N/A
[74]	MobileNetV1	98.14%	N/A	N/A

Table 6 Accuracy obtained on the Microsoft BIG 2015 dataset

Paper	Model	Accuracy	Precision	Recall
[56]	CNN	98.68%	N/A	N/A
[44]	RMVC	99.5%	N/A	N/A
[30]	CNN	97.4%	N/A	N/A
[61]	AIFS-IDL (GRU, TCN, and CNN)	99.92%	99.92%	99.92%
[31]	GAN + CNN	99.65%	96%	96%
[37]	MCSP and MCSLT	98.59%, 98.61%	N/A	N/A
[32]	CNN + ARM	99.95%, 99.63%	N/A	N/A
[33]	MalFCS	100%	N/A	N/A
[74]	MobileNetV1	98.95%	N/A	N/A

The advantage of applying the accuracy measure is that it is well defined, i.e., it can be calculated as the ratio of correct classifications and the total number of classifications. On the other hand, accuracy is not

necessarily an adequate measure in circumstances of imbalanced datasets. In general, a well-acknowledged way to overcome this evaluation problem is to apply the F1-score. Interestingly, it can be observed that the F1-score is applied less frequently (i.e., in 75,71 percent of papers, cf. Tab. 5) in the selected papers than accuracy. In addition, although the F1-score is well-acknowledged in machine learning, most of the considered papers do not specify how it is calculated. The problem with the F1-score is that it could be calculated in several different ways, as discussed in [112] for a general case of n-fold cross-validation:

The first approach is to calculate F1-scores for each fold, calculated for each class separately, and then the final F1-score is calculated as their average value:

Table 7 Accuracy obtained on the AndroZoo dataset

Paper	Model	Accuracy	Precision	Recall
[77]	ViT	80.27%	N/A	N/A
[29]	CNN	99.37%	N/A	N/A
[34]	AMalNet	99.69%	N/A	N/A

$$F_1 = \frac{1}{n}\sum_{i=1}^{n} F_{1,i} \tag{1}$$

The second approach is to average the precision and recall values across the folds separately, and then the final F1-score is calculated as follows:

$$P = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{2}$$

$$R = \frac{1}{n}\sum_{i=1}^{n} R_i \tag{3}$$

$$F_1 = \frac{2RP}{R+P} \tag{4}$$

The third approach is to count true positive, false positive, and true negative classifications across the folds, and then the final F1-score is calculated as follows:

$$TP = \frac{1}{n}\sum_{i=1}^{n} TP_i \tag{5}$$

$$FP = \frac{1}{n}\sum_{i=1}^{n} FP_i \tag{6}$$

$$FN = \frac{1}{n}\sum_{i=1}^{n} FN_i \tag{7}$$

$$F_1 = \frac{2TP}{2TP+FP+FN} \tag{8}$$

In [112], it is shown that the third approach is the most unbiased method in circumstances of imbalanced datasets. In the set of considered papers, cross-validation is relatively often applied (in 22 out of 70 papers). The application of the F1-score in these papers can be summarized as follows:

In 9 papers, the F1-score was not considered [19, 20, 27, 36, 41, 49, 51, 66, 71]. However, in [55] AUC was provided instead. The first approach to calculating the F1-score was applied in 3 papers [33, 46, 58]. The second approach to calculating the F1-score was applied in [32]. Simplified versions of this approach are applied in 9 additional papers [23, 34, 40, 54, 55, 59, 61-63]. The third approach to calculating the F1-score, which was recognized as the most unbiased, is not applied in the selected papers. Although the reported results sound promising, the sporadic absence of application of the F1-score and its inconsistent application do not allow for comparative analysis. We argue that researchers in this field should consistently apply the F1-score and clearly specify how it is calculated, especially in circumstances of imbalanced datasets and automatically applied software frameworks.

4.4 Study Limitations

To interpret the reported results, it is important to take into consideration the different limitations of the considered studies. In [56], the applied double-byte feature encoding method requires a large number of features and samples, and in some cases, there is an insufficient number of malware families and their variants [8]. On the other hand, in [58], the proposed approach does not consider structural knowledge and qualitative features of ELF binaries, such as strings or header fields. Additionally, representing a malware instance as a single grayscale image may be insufficient. Some papers [22] introduced a visualization approach that is not designed to address the obfuscation techniques. In contrast, there is a paper [29] where the proposed approach has not been tested for robustness against adversarial attacks. Also, an imbalanced dataset is considered a limitation [32]. In [81], there is only a brief examination of features such as the number of layers, neurons, and the impact of parameter values on the performance, while in [30], there is a lack of exploration into the performance of existing state-of-the-art deep learning systems using transfer learning. Additionally, in [33], there is an insufficient number of samples in the dataset, leading to overfitting. In [70], there is a potential lack of generalization to real-world malware samples, as the proposed approach has not been extensively tested on such data. Additionally, the time complexity of the system has not been thoroughly addressed, which may affect the classification efficiency for malicious samples. On the other hand, in [26], the considered features may not fully capture the malicious malware behavior. Additionally, the study may be constrained by the insufficient size of the dataset used for binary classification. Some papers highlight difficulties in classifying specific malware families [35]. Furthermore, in [82], the limitation of the study is its heavy reliance on static features, which may not fully capture dynamic behaviors, potentially resulting in

some malicious activities being overlooked. The authors simplified the call graph, introducing a risk of information loss in certain scenarios. The proposed model may encounter challenges related to the increased complexity and computational resource requirements when dealing with large-scale datasets. In [80], more diverse datasets are needed to prevent bias during sample selection, while in [61] proposed approach exhibits high computational time. In [68], the proposed approach does not consider the semantics of raw binary files. The spatial patterns inherent in each class of malware within the raw binary files remain unexploited, while in [46], the static detection process does not account for code obfuscation and encryption, potentially compromising the validity of the results, especially for detecting such types of malware. This limitation may hinder the effectiveness of detecting malware samples that do not primarily rely on network-based activities. Additionally, in [48], the generated datasets used to enhance model performance may contain irrelevant or problematic samples, thereby impacting the learning effectiveness of the model. The reported approach to screening adversarial samples during model training and verification may not fully exploit their potential to improve classification accuracy.

4 CONCLUSIONS

The conclusions can be summarized as follows:

(i) A malware conceptualization shift. The introduction of CNNs in static malware analysis did positively affect performance when compared to traditional, non-connectionist image processing approaches. The application of CNNs reflects the conceptualization of malware as a grid-like structure with local dependencies. Recently, this conceptualization has been extended, and malware was considered as a sequential structure with both local and long-term dependencies, which is reflected in the integration of recurrent neural networks in CNN-based architectures. This extended conceptualization is a step away from the fundamental observation that there are significant visual similarities in image texture for malware instances belonging to the same family (cf. [7]). Yet, there is still not a generalized conceptual model representation which would allow for a more complete understanding of the advantages and disadvantages of the selected approaches, as the resulting taxonomy does not give any causality.

(ii) The notion of dataset balance. Dataset imbalance is an inherently present problem. Unfortunately, the oversampling techniques are relatively frequently applied, although they are not necessarily methodologically justifiable for malware datasets. Even more importantly, it appears that the understanding of the notion of dataset balance should be revised in this field.

The typical understanding is that a dataset with equally represented classes should be considered representative. An alternative understanding, which is common in the field of corpus linguistics, assumes that the number of instances across classes should be proportional to their frequencies in real life. However, this alternative understanding is mostly overlooked in the considered papers (i.e., it was considered only in [44]).

(iii) Evaluation desideratum. There is much disagreement about whether the F1-score should be applied and ambiguity in how it should be calculated. Sometimes, the researchers appear unaware of different ways in which the F1-score can be calculated, e.g., they just apply a software framework to compute the F1-score. Taking these insights into account, we propose the following research directions in the field of CNN-based static malware analysis: The current conceptualization of malware as a sequential structure with both local and long-term dependencies does not integrate the information on the texture of the image representing a given malware instance. One of the possible future methodological directions aimed at improving the classification performance relates to the integration of techniques that are tailored to texture analysis with CNN-based architectures. The researchers should reconsider the notion of dataset balance. It is fair to assume that malware distribution in real life is not uniform. Thus, insisting on equally represented malware classes may introduce bias. Finally, to enable the interpretation and comparative analysis of evaluation results in the context of imbalanced datasets, researchers should consistently and transparently apply the F1-score. Finally, it should be noted that the considered studies take into account neither the issue of hallucination in convolutional neural networks, nor its potential effects on the vulnerability of the underlying models to adversarial attacks.

5 REFERENCES

[1] Alnajim, A. M., Habib, S., Islam, M., Thwin, S. M., & Alotaibi, F. (2023). A Comprehensive Survey of Cybersecurity Threats, Attacks, and Effective Countermeasures in Industrial Internet of Things. *Technologies*, *11*(6), 161. https://doi.org/10.3390/technologies11060161

[2] Pan, Y., Ge, X., Fang, C., & Fan, Y. (2020). A Systematic Literature Review of Android Malware Detection Using Static Analysis. *IEEE Access*, *8*, 116363-116379. https://doi.org/10.1109/ACCESS.2020.3002842

[3] Alqahtani, A., Azzony, S., Alsharafi, L., & Alaseri, M. (2023). Web-Based Malware Detection System Using Convolutional Neural Network. *Digital*, *3*(3), 273-285. https://doi.org/10.3390/digital3030017

[4] Komosar, A., Stefanović, D., & Sladojević, S. (2024). An overview of image processing in biomedicine using U-Net convolutional neural network architecture. *Journal of Computer and Forensic Sciences*, *00*, 4-4. https://doi.org/10.5937/jcfs3-48848

[5] Perić, D., Maček, N., & Bogdanoski, M. (2022). Application of convolutional neural networks to spoken words evaluation based on lip movements without accompanying sound signal. *Journal of Computer and Forensic Sciences*, *1*(1), 7-16. https://doi.org/10.5937/1-42696

[6] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

[7] Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011). Malware images: Visualization and automatic classification. *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, 1-7. https://doi.org/10.1145/2016904.2016908

[8] Almomani, I., Alkhayer, A., & El-Shafai, W. (2023). E2E-RDS: Efficient End-to-End Ransomware Detection System

Based on Static-Based ML and Vision-Based DL Approaches. *Sensors*, *23*(9), 4467. https://doi.org/10.3390/s23094467

[9] Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2024). A systematic literature review on Windows malware detection: Techniques, research issues, and future directions. *Journal of Systems and Software*, *209*, 111921. https://doi.org/10.1016/j.jss.2023.111921

[10] Senanayake, J., Kalutarage, H., & Al-Kadri, M. O. (2021). Android Mobile Malware Detection Using Machine Learning: A Systematic Review. *Electronics*, *10*(13), 1606. https://doi.org/10.3390/electronics10131606

[11] Ehsan, A., Catal, C., & Mishra, A. (2022). Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review. *Sensors*, *22*(20), 7928. https://doi.org/10.3390/s22207928

[12] Gaber, M. G., Ahmed, M., & Janicke, H. (2024). Malware Detection with Artificial Intelligence: A Systematic Literature Review. *ACM Computing Surveys*, *56*(6), 1-33. https://doi.org/10.1145/3638552

[13] Ketebu, K. E., Onwodi, G. O., Ukhurebor, K. E., Eneche, B. M., & Yaah-Nyakko, N. K. (2024). A recent survey of image-based malware classification using convolution neural network. *Journal of Autonomous Intelligence*, *7*(5), 1287. https://doi.org/10.32629/jai.v7i5.1287

[14] Sihwail, R., Omar, K., & Zainol Ariffin, K. A. (2018). A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis. *International Journal on Advanced Science, Engineering and Information Technology*, *8*(4-2), 1662-1671. https://doi.org/10.18517/ijaseit.8.4-2.6827

[15] Ali, R., Ali, A., Iqbal, F., Hussain, M., & Ullah, F. (2022). Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review. *Security and Communication Networks*, *2022*, 1-31. https://doi.org/10.1155/2022/2959222

[16] Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L. F., & Abdulkadir, S. J. (2022). Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review. *Electronics*, *11*(2), 198. https://doi.org/10.3390/electronics11020198

[17] Jaisinghani, K., & Singh, S. (n.d.). Recent Advances in Image based Malware Classification through the Lens of Deep Learning - A Systematic Literature Review. *International Journal of Intelligent Systems and Applications in Engineering.*

[18] Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, n71. https://doi.org/10.1136/bmj.n71

[19] Ding, Y., Hu, J., Xu, W., & Zhang, X. (2019). A Deep Feature Fusion Method for Android Malware Detection. *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, 1-6. https://doi.org/10.1109/ICMLC48188.2019.8949298

[20] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y., & Sakuma, J. (2019). Neural malware analysis with attention mechanism. *Computers & Security*, *87*, 101592. https://doi.org/10.1016/j.cose.2019.101592

[21] Xue, D., Li, J., Lv, T., Wu, W., & Wang, J. (2019). Malware Classification Using Probability Scoring and Machine Learning. *IEEE Access*, *7*, 91641-91656. https://doi.org/10.1109/ACCESS.2019.2927552

[22] Singh, A., Handa, A., Kumar, N., & Shukla, S. K. (2019). Malware Classification Using Image Representation. *Cyber Security Cryptography and Machine Learning*, *11527*, 75-92. https://doi.org/10.1007/978-3-030-20951-3_6

[23] Lim, H.-K., Kim, J.-B., Kim, K., Hong, Y.-G., & Han, Y.-H. (2019). Payload-Based Traffic Classification Using Multi-Layer LSTM in Software Defined Networks. *Applied Sciences*, *9*(12), 2550. https://doi.org/10.3390/app9122550

[24] Song, Y. & Wang, J. (2019). Efficient Shellcode Detection Based on Convolutional Neural Network. *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 309-313. https://doi.org/10.1109/CISCE.2019.00076

[25] Yen, Y.-S. & Sun, H.-M. (2019). An Android mutation malware detection based on deep learning using visualization of importance from codes. *Microelectronics Reliability*, *93*, 109-114. https://doi.org/10.1016/j.microrel.2019.01.007

[26] Wang, Z., Li, G., Chi, Y., Zhang, J., Yang, T., & Liu, Q. (2019). Android Malware Detection Based on Convolutional Neural Networks. *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 1-6. https://doi.org/10.1145/3331453.3361306

[27] Jeong, Y.-S., Woo, J., & Kang, A. R. (2019). Malware Detection on Byte Streams of Hangul Word Processor Files. *Applied Sciences*, *9*(23), 5178. https://doi.org/10.3390/app9235178

[28] Lee, Y., Kwon, H., Choi, S.-H., Lim, S.-H., Baek, S. H., & Park, K.-W. (2019). Instruction2vec: Efficient Preprocessor of Assembly Code to Detect Software Weakness with CNN. *Applied Sciences*, *9*(19), 4086. https://doi.org/10.3390/app9194086

[29] Darwaish, A. & Nait-Abdesselam, F. (2020). RGB-based Android Malware Detection and Classification Using Convolutional Neural Network. *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 1-6. https://doi.org/10.1109/GLOBECOM42002.2020.9348206

[30] Sartoli, S., Wei, Y., & Hampton, S. (2020). Malware Classification using Recurrence Plots and Deep Neural Network. *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 901-906. https://doi.org/10.1109/ICMLA51294.2020.00147

[31] Jang, S., Li, S., & Sung, Y. (2020). Generative Adversarial Network for Global Image-Based Local Image to Improve Malware Classification Using Convolutional Neural Network. *Applied Sciences*, *10*(21), 7585. https://doi.org/10.3390/app10217585

[32] Tariang, D. B., Birudaraju, S. C., Naskar, R., Khare, V., & Chakraborty, R. S. (2020). Malware Classification Through Attention Residual Network based Visualization. *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 1-6. https://doi.org/10.1109/AsianHOST51057.2020.9358249

[33] Xiao, G., Li, J., Chen, Y., & Li, K. (2020). MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *Journal of Parallel and Distributed Computing*, *141*, 49-58. https://doi.org/10.1016/j.jpdc.2020.03.012

[34] Pei, X., Yu, L., & Tian, S. (2020). AMalNet: A deep learning framework based on graph convolutional networks for malware detection. *Computers & Security*, *93*, 101792. https://doi.org/10.1016/j.cose.2020.101792

[35] Brezinski, K. & Ferens, K. (2020). Complexity-Based Convolutional Neural Network for Malware Classification. *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, 1-9. https://doi.org/10.1109/CSCI51800.2020.00008

[36] Nisa, M., Shah, J. H., Kanwal, S., Raza, M., Khan, M. A., Damaševičius, R., & Blažauskas, T. (2020). Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features. *Applied Sciences*, *10*(14), 4966.

https://doi.org/10.3390/app10144966
[37] Kwon, Y.-M., An, J.-J., Lim, M.-J., Cho, S., & Gal, W.-M. (2020). Malware Classification Using Simhash Encoding and PCA (MCSP). *Symmetry*, *12*(5), 830. https://doi.org/10.3390/sym12050830
[38] Jeong, Y.-S., Woo, J., Lee, S., & Kang, A. R. (2020). Malware Detection of Hangul Word Processor Files Using Spatial Pyramid Average Pooling. *Sensors*, *20*(18), 5265. https://doi.org/10.3390/s20185265
[39] Wang, L., Li, X., Wang, R., Xin, Y., Gao, M., & Chen, Y. (2020). PreNNsem: A Heterogeneous Ensemble Learning Framework for Vulnerability Detection in Software. *Applied Sciences*, *10*(22), 7954. https://doi.org/10.3390/app10227954
[40] Choi, S. (2020). Malicious PowerShell Detection Using Attention against Adversarial Attacks. *Electronics*, *9*(11), 1817. https://doi.org/10.3390/electronics9111817
[41] Choi, S., Bae, J., Lee, C., Kim, Y., & Kim, J. (2020). Attention-Based Automated Feature Extraction for Malware Analysis. *Sensors*, *20*(10), 2893. https://doi.org/10.3390/s20102893
[42] Thosar, K., Tiwari, P., Jyothula, R., & Ambawade, D. (2021). Effective Malware Detection using Gradient Boosting and Convolutional Neural Network. *2021 IEEE Bombay Section Signature Conference (IBSSC)*, 1-4. https://doi.org/10.1109/IBSSC53889.2021.9673266
[43] Kural, O. E., Sahin, D. O., Akleylek, S., Kilic, E., & Omural, M. (2021). Apk2Img4AndMal: Android Malware Detection Framework Based on Convolutional Neural Network. *2021 6th International Conference on Computer Science and Engineering (UBMK)*, 731-734. https://doi.org/10.1109/UBMK52708.2021.9558983
[44] Sun, G. & Qian, Q. (2021). Deep Learning and Visualization for Identifying Malware Families. *IEEE Transactions on Dependable and Secure Computing*, *18*(1), 283-295. https://doi.org/10.1109/TDSC.2018.2884928
[45] Awan, M. J., Masood, O. A., Mohammed, M. A., Yasin, A., Zain, A. M., Damaševičius, R., & Abdulkareem, K. H. (2021). Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention. *Electronics*, *10*(19), 2444. https://doi.org/10.3390/electronics10192444
[46] Ding, C., Luktarhan, N., Lu, B., & Zhang, W. (2021). A Hybrid Analysis-Based Approach to Android Malware Family Classification. *Entropy*, *23*(8), 1009. https://doi.org/10.3390/e23081009
[47] El-Shafai, W., Almomani, I., & AlKhayer, A. (2021). Visualized Malware Multi-Classification Framework Using Fine-Tuned CNN-Based Transfer Learning Models. *Applied Sciences*, *11*(14), 6446. https://doi.org/10.3390/app11146446
[48] Wang, C., Zhang, L., Zhao, K., Ding, X., & Wang, X. (2021). AdvAndMal: Adversarial Training for Android Malware Detection and Family Classification. *Symmetry*, *13*(6), 1081. https://doi.org/10.3390/sym13061081
[49] Yang, Y., Du, X., Yang, Z., & Liu, X. (2021). Android Malware Detection Based on Structural Features of the Function Call Graph. *Electronics*, *10*(2), 186. https://doi.org/10.3390/electronics10020186
[50] Zhou, X. (2021). Homology Detection of Malicious Codes Based on a Fuzzy Graph Neural Network. *2021 IEEE International Conference on Industrial Application of Artificial Intelligence (IAAI)*, 202-207. https://doi.org/10.1109/IAAI54625.2021.9699879
[51] Hamad, S. A., Tran, D. H., Sheng, Q. Z., & Zhang, W. E. (2021). BERTDeep-Ware: A Cross-architecture Malware Detection Solution for IoT Systems. *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 927-934. https://doi.org/10.1109/TrustCom53373.2021.00130
[52] Coleman, S.-P. W., & Hwang, Y.-S. (2021). Android Malware Detection System using Deep Learning and Code Item. *IEIE Transactions on Smart Processing & Computing*, *10*(2), 116-121. https://doi.org/10.5573/IEIESPC.2021.10.2.116
[53] Zhang, N., Xue, J., Ma, Y., Zhang, R., Liang, T., & Tan, Y. (2021). Hybrid sequence-based Android malware detection using natural language processing. *International Journal of Intelligent Systems*, *36*(10), 5770-5784. https://doi.org/10.1002/int.22529
[54] Ashik, M., Jyothish, A., Anandaram, S., Vinod, P., Mercaldo, F., Martinelli, F., & Santone, A. (2021). Detection of Malicious Software by Analyzing Distinct Artifacts Using Machine Learning and Deep Learning Algorithms. *Electronics*, *10*(14), 1694. https://doi.org/10.3390/electronics10141694
[55] Zhang, W., Luktarhan, N., Ding, C., & Lu, B. (2021). Android Malware Detection Using TCN with Bytecode Image. *Symmetry*, *13*(7), 1107. https://doi.org/10.3390/sym13071107
[56] Li, L., Ding, Y., Li, B., Qiao, M., & Ye, B. (2022). Malware classification based on double byte feature encoding. *Alexandria Engineering Journal*, *61*(1), 91-99. https://doi.org/10.1016/j.aej.2021.04.076
[57] Shah, S. S. H., Jamil, N., & Khan, A. U. R. (2022). Performance comparison of visualization-based malware detection and classification techniques. *2022 17th International Conference on Emerging Technologies (ICET)*, 200-205. https://doi.org/10.1109/ICET56601.2022.10004652
[58] Belguendouz, H., Guerid, H., & Kaddour, M. (2022). Static Classification of IoT Malware using Grayscale Image Representation and Lightweight Convolutional Neural Networks. *2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet)*, 1-8. https://doi.org/10.1109/CommNet56067.2022.9993956
[59] Amenova, S., Turan, C., & Zharkynbek, D. (2022). Android Malware Classification by CNN-LSTM. *2022 International Conference on Smart Information Systems and Technologies (SIST)*, 1-4. https://doi.org/10.1109/SIST54437.2022.9945816
[60] Lin, C.-J., Huang, M.-S., & Lee, C.-L. (2022). Malware Classification Using Convolutional Fuzzy Neural Networks Based on Feature Fusion and the Taguchi Method. *Applied Sciences*, *12*(24), 12937. https://doi.org/10.3390/app122412937
[61] Wu, X. & Song, Y. (2022). An Efficient Malware Classification Method Based on the AIFS-IDL and Multi-Feature Fusion. *Information*, *13*(12), 571. https://doi.org/10.3390/info13120571
[62] Wu, X., Song, Y., Hou, X., Ma, Z., & Chen, C. (2022). Deep Learning Model with Sequential Features for Malware Classification. *Applied Sciences*, *12*(19), 9994. https://doi.org/10.3390/app12199994
[63] Ben Atitallah, S., Driss, M., & Almomani, I. (2022). A Novel Detection and Multi-Classification Approach for IoT-Malware Using Random Forest Voting of Fine-Tuning Convolutional Neural Networks. *Sensors*, *22*(11), 4302. https://doi.org/10.3390/s22114302
[64] Obaidat, I., Sridhar, M., Pham, K. M., & Phung, P. H. (2022). Jadeite: A novel image-behavior-based approach for Java malware detection using deep learning. *Computers & Security*, *113*, 102547. https://doi.org/10.1016/j.cose.2021.102547
[65] Ullah, F., Alsirhani, A., Alshahrani, M. M., Alomari, A., Naeem, H., & Shah, S. A. (2022). Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation. *Sensors*, *22*(18), 6766. https://doi.org/10.3390/s22186766
[66] Dener, M., Ok, G., & Orman, A. (2022). Malware Detection Using Memory Analysis Data in Big Data Environment. *Applied Sciences*, *12*(17), 8604. https://doi.org/10.3390/app12178604

[67] Peng, T., Hu, B., Liu, J., Huang, J., Zhang, Z., He, R., & Hu, X. (2022). A Lightweight Multi-Source Fast Android Malware Detection Model. *Applied Sciences*, *12*(11), 5394. https://doi.org/10.3390/app12115394

[68] Akhtar, M. S. & Feng, T. (2022). Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time. *Symmetry*, *14*(11), 2308. https://doi.org/10.3390/sym14112308

[69] Ngo, M. V., Truong-Huu, T., Rabadi, D., Loo, J. Y., & Teo, S. G. (2023). Fast and Efficient Malware Detection with Joint Static and Dynamic Features Through Transfer Learning. *Applied Cryptography and Network Security*, *13905*, 503-531. https://doi.org/10.1007/978-3-031-33488-7_19

[70] Chen, Z. & Cao, J. (2023). VMCTE: Visualization-Based Malware Classification Using Transfer and Ensemble Learning. *Computers, Materials & Continua*, *75*(2), 4445-4465. https://doi.org/10.32604/cmc.2023.038639

[71] Shaukat, K., Luo, S., & Varadharajan, V. (2023). A novel deep learning-based approach for malware detection. *Engineering Applications of Artificial Intelligence*, *122*, 106030. https://doi.org/10.1016/j.engappai.2023.106030

[72] Cha, H.-J., Yang, H.-K., Song, Y.-J., & Kang, A. R. (2023). Intelligent Anomaly Detection System through Malware Image Augmentation in IIoT Environment Based on Digital Twin. *Applied Sciences*, *13*(18), 10196. https://doi.org/10.3390/app131810196

[73] Huang, H., Du, R., Wang, Z., Li, X., & Yuan, G. (2023). A Malicious Code Detection Method Based on Stacked Depthwise Separable Convolutions and Attention Mechanism. *Sensors*, *23*(16), 7084. https://doi.org/10.3390/s23167084

[74] Alnajim, A. M., Habib, S., Islam, M., Albelaihi, R., & Alabdulatif, A. (2023). Mitigating the Risks of Malware Attacks with Deep Learning Techniques. *Electronics*, *12*(14), 3166. https://doi.org/10.3390/electronics12143166

[75] Taher, F., AlFandi, O., Al-kfairy, M., Al Hamadi, H., & Alrabaee, S. (2023). DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection. *Applied Sciences*, *13*(13), 7720. https://doi.org/10.3390/app13137720

[76] Zhu, J., Jang-Jaccard, J., Singh, A., Watters, P. A., & Camtepe, S. (2023). Task-Aware Meta Learning-Based Siamese Neural Network for Classifying Control Flow Obfuscated Malware. *Future Internet*, *15*(6), 214. https://doi.org/10.3390/fi15060214

[77] Jo, J., Cho, J., & Moon, J. (2023). A Malware Detection and Extraction Method for the Related Information Using the ViT Attention Mechanism on Android Operating System. *Applied Sciences*, *13*(11), 6839. https://doi.org/10.3390/app13116839

[78] Wu, H., Luktarhan, N., Tian, G., & Song, Y. (2023). An Android Malware Detection Approach to Enhance Node Feature Differences in a Function Call Graph Based on GCNs. *Sensors*, *23*(10), 4729. https://doi.org/10.3390/s23104729

[79] Panda, P., C U, O. K., Marappan, S., Ma, S., S, M., & Veesani Nandi, D. (2023). Transfer Learning for Image-Based Malware Detection for IoT. *Sensors*, *23*(6), 3253. https://doi.org/10.3390/s23063253

[80] Da Silva, A. A. & Pamplona Segundo, M. (2023). On Deceiving Malware Classification with Section Injection. *Machine Learning and Knowledge Extraction*, *5*(1), 144-168. https://doi.org/10.3390/make5010009

[81] Calik Bayazit, E. (2023). Deep Learning based Malware Detection for Android Systems: A Comparative Analysis. *Tehnicki Vjesnik - Technical Gazette*, *30*(3). https://doi.org/10.17559/TV-20220907113227

[82] Xu, Q., Zhao, D., Yang, S., Xu, L., & Li, X. (2023). Android Malware Detection Based on Behavioral-Level Features with Graph Convolutional Networks. *Electronics*, *12*(23), 4817. https://doi.org/10.3390/electronics12234817

[83] Almazroi, A. A. & Ayub, N. (2023). Enhancing Smart IoT Malware Detection: A GhostNet-based Hybrid Approach. *Systems*, *11*(11), 547. https://doi.org/10.3390/systems11110547

[84] Rahali, A. & Akhloufi, M. A. (2023). MalBERTv2: Code Aware BERT-Based Model for Malware Identification. *Big Data and Cognitive Computing*, *7*(2), 60. https://doi.org/10.3390/bdcc7020060

[85] Taher, F., Al Fandi, O., Al Kfairy, M., Al Hamadi, H., & Alrabaee, S. (2023). A Proposed Artificial Intelligence Model for Android-Malware Detection. *Informatics*, *10*(3), 67. https://doi.org/10.3390/informatics10030067

[86] Jonnala, Y. D., Mahajan, V. S., Menon, D., Kothakapu, S. R., & Chandamollu, S. R. (2023). Malware Detection Using Binary Visualization and Neural Networks. *E3S Web of Conferences*, *391*, 01107. https://doi.org/10.1051/e3sconf/202339101107

[87] McEnery, T., Xiao, R., & Tono, Y. (2005). *Corpus-based language studies: An advanced resource book* (Reprinted). Routledge

[88] Allix, K., Bissyandé, T. F., Klein, J., & Le Traon, Y. (2016). AndroZoo: Collecting millions of Android apps for the research community. *Proceedings of the 13th International Conference on Mining Software Repositories*, 468-471. https://doi.org/10.1145/2901739.2903508

[89] Anderson, H. S. & Roth, P. (2018). *EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models.* https://doi.org/10.48550/ARXIV.1804.04637

[90] Jang, J., Kang, H., Woo, J., Mohaisen, A., & Kim, H. K. (2016). Andro-Dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information. *Computers & Security*, *58*, 125-138. https://doi.org/10.1016/j.cose.2015.12.005

[91] Abdul Kadir, A. F., Stakhanova, N., & Ghorbani, A. A. (2015). Android Botnets: What URLs are Telling Us. *Network and System Security*, *9408*, 78-91. https://doi.org/10.1007/978-3-319-25645-0_6

[92] Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., & Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. *Proceedings 2014 Network and Distributed System Security Symposium*. https://doi.org/10.14722/ndss.2014.23247

[93] *Android Malware Genome Project*. (n.d.) Retrieved from http://www.malgenomeproject.org/

[94] Miranda, T. C., Gimenez, P.-F., Lalande, J.-F., Tong, V. V. T., & Wilke, P. (2022). Debiasing Android Malware Datasets: How Can I Trust Your Results if Your Dataset is Biased? *IEEE Transactions on Information Forensics and Security*, *17*, 2182-2197. https://doi.org/10.1109/TIFS.2022.3180184

[95] *Microsoft Malware Classification Challenge (BIG 2015)*. Retrieved from https://kaggle.com/competitions/malware-classification

[96] Mahdavifar, S., Abdul Kadir, A. F., Fatemi, R., Alhadidi, D., & Ghorbani, A. A. (2020). Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning. *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 515-522.

[97] Lashkari, A. H., Kadir, A. F. A., Taheri, L., & Ghorbani, A. A. (2018). Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. *2018 International Carnahan Conference on Security Technology (ICCST)*, 1-7. https://doi.org/10.1109/CCST.2018.8585560

[98] Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019). Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls. *2019*

International Carnahan Conference on Security Technology (ICCST), 1-8. https://doi.org/10.1109/CCST.2019.8888430

[99] Carrier, T., Victor, P., Tekeoglu, A., & Lashkari, A. (2022). Detecting Obfuscated Malware using Memory Feature Engineering: *Proceedings of the 8th International Conference on Information Systems Security and Privacy*, 177-188. https://doi.org/10.5220/0010908200003120

[100] Polychronakis, M. & Meier, M. (Eds.). (2017). *Detection of Intrusions and Malware, and Vulnerability Assessment: 14th International Conference, DIMVA 2017, Bonn, Germany, July 6-7, 2017, Proceedings*, *10327*. https://doi.org/10.1007/978-3-319-60876-1

[101] Kumar, A. (2020). *ClaMP (Classification of Malware with PE headers)* [Dataset]. https://doi.org/10.17632/XVYV59VWVZ.1

[102] *Benign & Malicious PE Files*. (n.d.) Retrieved from https://www.kaggle.com/datasets/amauricio/pe-files-malwares

[103] Yıldırım, E. (2024). *Emr4h/Malware-Detection-Using-Machine-Learning* [Jupyter Notebook]. https://github.com/emr4h/Malware-Detection-Using-Machine-Learning (Original work published 2022)

[104] Almomani, I., Qaddoura, R., Habib, M., Alsoghyer, S., Khayer, A. A., Aljarah, I., & Faris, H. (2021). Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data. *IEEE Access*, *9*, 57674-57691. https://doi.org/10.1109/ACCESS.2021.3071450

[105] Bozkir, A. S., Tahillioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*, *103*, 102166. https://doi.org/10.1016/j.cose.2020.102166

[106] Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). *IoT-23: A labeled dataset with malicious and benign IoT network traffic* (Version 1.0.0) [Dataset]. https://doi.org/10.5281/ZENODO.4743746

[107] Mariconti, E., Onwuzurike, L., Andriotis, P., De Cristofaro, E., Ross, G., & Stringhini, G. (2017). MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. *Proceedings 2017 Network and Distributed System Security Symposium*. https://doi.org/10.14722/ndss.2017.23353

[108] *MaleVis Dataset*. (n.d.) Retrieved from https://www.kaggle.com/datasets/nimit5/malevis-dataset

[109] External Data Source. (2018). *Android PRAGuard Dataset* [Dataset]. https://doi.org/10.23721/100/1504381

[110] Hossin, M. & Sulaiman, M. N. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*(2), 01-11. https://doi.org/10.5121/ijdkp.2015.5201

[111] Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), 6086. https://doi.org/10.1038/s41598-024-56706-x

[112] Forman, G. & Scholz, M. (2010). Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, *12*(1), 49-57. https://doi.org/10.1145/1882471.1882479

Contact information:

Aleksa KOMOSAR
Faculty of Technical Sciences, University of Novi Sad,
Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
E-mail: aleksakomosar@uns.ac.rs

Milan GNJATOVIC
(Corresponding author)
Department of Information Technology,
University of Criminal Investigation and Police Studies,
Cara Dušana 196, 11080 Belgrade, Serbia
E-mail: milan.gnjatovic@kpu.edu.rs

Darko STEFANOVIC
Faculty of Technical Sciences, University of Novi Sad,
Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
E-mail: darko.stefanovic@uns.ac.rs

Nemanja MACEK
Academy of Technical and Art Applied Studies,
School of Electrical and Computer Engineering,
Vojvode Stepe 283, 11000 Beograd, Serbia
E-mail: nmacek@viser.edu.rs

Dusan SAVIC
Faculty of Organizational Sciences, University of Belgrade,
Jove Ilića 154, 11010 Belgrade, Serbia
E-mail: savic.dusan@fon.bg.ac.rs

Teodora VUCKOVIC
Faculty of Technical Sciences, University of Novi Sad,
Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia
E-mail: teodora.lolic@uns.ac.rs