И

УДК 519.71 **DOI** 10.24412/2413-7383- 106-122

Y. Imamverdiyev, E. Baghirov, I.J. Chukwu
Azerbaijan Technical University, H. Javid 25, Baku, Azerbaijan
Institute of Information Technology of the Ministry of Science and Education, B. Vahabzade 9A, Baku, Azerbaijan
Kadir Has University, Istanbul, Türkiye, and Ss. Cyril and Methodius University in Skopje (UKIM), North Macedonia

IMAGE-BASED DEEP LEARNING METHOD FOR EFFECTIVE MALWARE DETECTION

Я. Имамвердиев[1], Э. Багиров[2], И.Дж. Чукву[3]
[1]Азербайджанский Технический Университет, ул. Х. Джавида, 25, Баку, Азербайджан
[2]Институт Информационных Технологий Министерства Науки и Образования, ул. Б. Вахабзаде, 9А, Баку, Азербайджан
[3]Университет Кадир Хас, Стамбул, Турция, и Университет им. Св. Кирилла и Мефодия (UKIM), Скопье, Северная Македония

МЕТОД ГЛУБОКОГО ОБУЧЕНИЯ НА ОСНОВЕ ИЗОБРАЖЕНИЙ ДЛЯ ЭФФЕКТИВНОГО ОБНАРУЖЕНИЯ ВРЕДОНОСНОГО ПО

The article examines a method for malware detection based on the analysis of grayscale images. Thirteen advanced convolutional neural networks, including DenseNet201, MobileNet, and others, are utilized for analysis based on the Malimg dataset. Experiments were conducted, including training and hyperparameter tuning, to optimize the models' performance. It is shown that models such as DenseNet201 and MobileNet achieve high accuracy, precision, recall, and F1 scores. This approach enhances the malware detection process, ensuring high efficiency and resilience against traditional methods of bypassing security systems. The application area of this work includes modern cybersecurity systems, the development of new methods for malware analysis, and protection against cyberattacks.

Key words: Malware detection, malware analysis, transfer learning, image-based detection, cybersecurity

В статье рассматривается метод обнаружения вредоносного программного обеспечения на основе анализа изображений, представленных в оттенках серого. Для анализа используются 13 современных сверточных нейронных сетей, включая DenseNet201, MobileNet, и другие, на основе набора данных Malimg. Проведены эксперименты, включающие обучение и настройку гиперпараметров для оптимизации производительности моделей. Показано, что модели, такие как DenseNet201 и MobileNet, достигают высокой точности, полноты, точности и F1-метрики. Данный подход позволяет улучшить процесс обнаружения вредоносного ПО, обеспечивая высокую эффективность и устойчивость к традиционным методам обхода систем защиты. Область применения работы — современные системы кибербезопасности, включая разработку новых методов анализа вредоносного ПО и защиту от кибератак.

Ключевые слова: Обнаружение вредоносного ПО, анализ вредоносного ПО, перенос обучения, обнаружение на основе изображений, кибербезопасность

Introduction

Malware, short for malicious software, refers to any software designed to disrupt, damage, or gain unauthorized access to computer systems [1]. As cyber threats continue to evolve, malware has become increasingly sophisticated and diverse, posing significant challenges to cybersecurity. The ongoing battle between malware distributors and the extensive community dedicated to malware detection remains intense [2]. Cybercriminals develop malware due to its substantial destructive capabilities.

Malware detection methods can be divided into three main categories [3]. Signature-based detection is one of the earliest and most widely used methods for identifying malware. It works by scanning files for specific patterns or signatures characteristic of known malware [4]. When a match is found, the file is flagged as malicious. While effective for known threats, this method is limited by its reliance on an up-to-date database of malware signatures. It fails to detect new, unknown malware (zero-day attacks) and malware that employs polymorphic and metamorphic techniques to alter its signature with each infection [1].

Dynamic analysis, or behavioral analysis, involves executing the suspicious file in a controlled environment (sandbox) and observing its behavior. This method focuses on the actions performed by the software, such as file modifications, network communications, and interactions with the operating system. Dynamic analysis can detect malware that employs obfuscation techniques to evade static analysis. However, it is more resource-intensive and time-consuming, requiring a secure environment to execute and monitor the software [1].

Hybrid analysis combines the strengths of both static and dynamic analysis to provide a more comprehensive approach to malware detection. By integrating static analysis's speed and automation with dynamic analysis's detailed behavioral insights, hybrid analysis can detect a broader range of malware, including those that evade one type of analysis alone. This approach helps to improve detection accuracy and reduce false positives, making it a robust solution for modern malware threats. Implementing a hybrid analysis system can be quite complex due to the need to integrate static and dynamic analysis tools seamlessly [5].

In recent years, numerous works have applied deep learning or NLP techniques for malware detection [1, 6–8]. This innovative approach involves transforming malware binaries into images and analyzing these images using convolutional neural networks (CNNs). The rationale behind this method is that the binary structure of malware can reveal unique patterns and features when visualized, which can be effectively captured and classified by CNNs.

Despite the advancements in malware detection techniques, several challenges remain. Signature-based detection methods are limited by their inability to identify new, unknown malware and those that employ sophisticated evasion techniques. While effective in uncovering hidden behaviors, dynamic analysis is resource-intensive and not scalable for large-scale real-time protection. Although promising, hybrid analysis faces complexities in seamlessly integrating static and dynamic methods. Moreover, the increasing sophistication of malware, including the use of advanced obfuscation and encryption methods, continues to outpace traditional detection mechanisms, necessitating the development of more advanced and adaptive approaches.

Hybrid analysis and dynamic analysis of malware typically require executing it in a controlled environment to observe its behavior, which can be resource-intensive and risky. Static analysis, on the other hand, involves disassembling the malware code to inspect its structure without executing it, but advanced obfuscation techniques can thwart this approach. Image-based malware detection, however, offers a distinct advantage as it does not

necessitate executing or disassembling the malware. Instead, it transforms malware binaries into images, leveraging deep learning models to detect malicious patterns, thus providing a safer and often more efficient alternative for malware analysis.

The main contributions of this work are listed below:

– The application of 13 pre-trained transfer learning models for enhanced malware detection, demonstrating the effectiveness of image-based deep learning techniques.

– A detailed performance comparison of these models using metrics such as accuracy, precision, recall, and F1-score, along with accuracy plots per epoch.

– An extensive evaluation of other authors' works using the same Malimg dataset showing that the proposed models outperform existing approaches.

The remainder of this paper is organized as follows: "Literature review" section provides a comprehensive review of related works in malware detection, mainly focusing on imagebased approaches. "Materials and methodologies" section outlines dataset descriptions, transfer learning models, image conversion processes, and evaluation metrics. "Experiments and results" section covering data preprocessing and the results of the experiments. Finally, "Conclusion" section concludes the paper by summarizing the essential findings and discussing the limitations and potential directions for future research.

Literature review

DEXRAY [9] proposed a method that converts the byte-code of DEX files from the Android app into grayscale images and processes them with a 1-dimensional convolutional neural network (CNN), demonstrating high detection effectiveness on a dataset of over 158,000 apps. The study emphasizes the potential of image-based deep learning for malware detection, providing a simple yet effective baseline for future research in this domain. Additionally, the impact of time decay and image-resizing on DEXRAY's performance and its resilience to obfuscation were investigated. This work contributes to the field by offering a foundational approach that can guide further exploration and development in deep learningbased malware detection. The simplicity of its design choices, intended to establish a baseline, may limit its ability to capture complex patterns in malware behavior, potentially leading to lower detection accuracy compared to more sophisticated models that use advanced feature extraction and data augmentation techniques. Another concern is the model's resilience to adversarial attacks and sophisticated obfuscation techniques, which needs further investigation as malware developers often use advanced methods to evade detection.

To this effect, [10] suggested a novel static malware detection method utilizing an enhanced AlexNet convolutional neural network (CNN) to address the inefficiencies of traditional machine learning and dynamic analysis methods in classifying large quantities of malware. By converting malware bytes into color images and improving the AlexNet architecture, the study aims to extract the texture features of malware better. A data enhancement method is employed to tackle unbalanced datasets. Extensive experiments using the Microsoft malware dataset and the Google Code Jam (GCJ) dataset demonstrate high accuracy, achieving 99.99% for the Microsoft dataset and 99.38% for the GCJ dataset. The results indicate significant improvements in both accuracy and detection efficiency. The approach involves many model parameters, which can complicate the model's implementation and increase computational requirements. Additionally, the method currently analyzes only one sample feature, potentially limiting its effectiveness. The authors suggest that future work will address these limitations by extracting more features, such as entropy images and APIs, and combining dynamic and static analysis. Moreover, they plan

to conduct experiments using real-world malware datasets to validate and enhance the proposed method's performance.

[11] combines transfer learning and deep convolutional neural network (CNN) models to enhance malware detection. Initially, the pre-trained VGG16 model is fine-tuned with various hyperparameters and used as a feature extractor. Alongside VGG16, three other pre-trained CNN models—VGG19, ResNet50, and InceptionV3—are utilized to extract diverse feature maps. These features are concatenated to form a comprehensive feature map, which undergoes feature selection processes to eliminate irrelevant data. The study then employs six classifiers K-Nearest Neighbor (K-NN), Support Vector Machine (SVM), Random Forest (RF), Multi-Layer Perceptron (MLP), Extra Tree (ET), and Gaussian Naive Bayes (GNB) using the stacked feature map as the training vector. The MLP model is further optimized using a randomized search algorithm, achieving the best performance with 98.55% accuracy, 99% precision, 99% recall, and 99% F1-score on the MalImg dataset, and 94.78% accuracy on real-world malware datasets. This method proves effective against standard obfuscation techniques without requiring code disassembly or dynamic analysis. Integrating more advanced feature selection techniques and evaluating the system's performance against emerging malware threats could further enhance its robustness and applicability.

The literature review reveals that image-based deep learning approaches are increasingly utilized for malware detection and classification, demonstrating significant potential in this field. These methodologies typically involve converting malware binaries into visual representations, such as grey-scale or color images, which are then processed using advanced convolutional neural networks (CNNs).

Despite their promise, several common challenges persist across these studies. These include the high computational complexity associated with fine-tuning pretrained models, the large number of model parameters, and the limited scope of feature analysis. Additionally, while some methods show resilience to standard obfuscation techniques, the effectiveness of these approaches against sophisticated obfuscation and adversarial attacks remains a critical area for further investigation.

Materials and methodologies

This section outlines the technologies used and provides the background information necessary to comprehend the proposed methodology.

Transfer Learning. Transfer learning has become an indispensable technique in modern deep learning, particularly for image classification tasks. It involves utilizing models pretrained on large-scale datasets, such as ImageNet, which contains millions of labeled images across thousands of categories. The pretraining process allows these models to learn a wide variety of features, from simple edges and textures in the early layers to more complex patterns and object parts in the deeper layers [12].

The core idea behind transfer learning is to leverage the knowledge acquired by these pre-trained models to improve performance on new tasks with limited data. Instead of training a neural network from scratch, which requires substantial computational resources and time, transfer learning allows for adapting pretrained models to a new, smaller dataset. This adaptation can involve either training the entire network further on the latest data or using the pre-trained model as a fixed feature extractor and training only the final classification layer.

Several well-known pre-trained models are widely adopted for transfer learning in various image recognition tasks. These include:

VGG16 and VGG19: Developed by the Visual Geometry Group (VGG) at the University of Oxford, these models are known for their simplicity and depth. They consist of 16 and 19 layers, respectively, and use small receptive fields (3x3 convolutional filters) throughout the network. Their consistent performance has made them popular choices for many computer vision tasks [12, 13].

ResNet (Residual Networks): Introduced by Microsoft Research, ResNet models, such as ResNet50, ResNet101, and ResNet152, utilize residual learning to tackle the vanishing gradient problem in intense networks. These models include shortcut connections that skip one or more layers, allowing gradients to flow directly through the network. This enables the training of much deeper architectures without performance degradation [14].

MobileNet and MobileNetV2: Developed by Google, these models are designed to be efficient and lightweight, making them suitable for mobile and embedded vision applications. They use depthwise separable convolutions to reduce the number of parameters and computational complexity while maintaining high accuracy [15–17].

DenseNet (Densely Connected Convolutional Networks): Proposed by researchers at the University of California, Berkeley, DenseNet models, such as DenseNet201, connect each layer to every other layer in a feed-forward fashion. This connectivity pattern helps mitigate the vanishing gradient problem, strengthens feature propagation, and enables efficient parameter usage [18].

Xception: An extension of the Inception architecture, Xception (Extreme Inception) replaces the starobustception modules with depthwise separable convolutions. This model, proposed by François Chollet, achieves superior performance by mapping cross-channel and spatial correlations independently [19].

InceptionResNetV2: Combining the strengths of Inception modules and residual connections, this model integrates the architecture of Inception with the residual connections of ResNet, resulting in a highly efficient and powerful network [20].

EfficientNet: Developed by Google AI, EfficientNet models utilize a compound scaling method that uniformly scales all dimensions of depth, width, and resolution using a fixed set of scaling coefficients. This approach achieves state-of-the-art accuracy while being computationally efficient [21].

NASNetLarge: The Neural Architecture Search Network (NASNet), developed by Google, uses reinforcement learning to automate the design of neural network architectures. NASNetLarge represents a highperforming model discovered through this automated search process [22].

These pre-trained models provide a robust foundation for transfer learning, enabling researchers and practitioners to perform highly on specific tasks with relatively little additional training. Their widespread adoption and proven efficacy make them indispensable tools in the modern deep-learning toolkit. The specifications of the various pre-trained models used in this study are detailed in Table 1.

The overall workflow of the proposed method is depicted in Figure 1. The first step involves converting the binary files into images, as detailed in 3.3. After obtaining the images, several data preprocessing steps are applied to prepare the dataset. The prepared images are then split into training, testing, and validation sets. Thirteen pretrained models are applied to the training dataset and evaluated on the test and validation datasets to assess overfitting and model robustness. Optimizer parameters, including Adam and SGD, are tested across all models.

Table 1: Specifications of Various Pretrained Models

Model	Depth (Layers)	Parameters (Millions)	Architecture Type	Unique Features
ResNet50	50	25	Residual	Residual connections
ResNet101	101	45	Residual	Residual connections
ResNet152	152	60	Residual	Residual connections
Xception	71	22	Depthwise Separable Conv	Extreme Inception with separable convolutions
MobileNet	28	4.2	Depthwise Separable Conv	Efficient for mobile and embedded systems
MobileNetV2	53	3.4	Depthwise Separable Conv	Inverted Residuals and Linear Bottlenecks
VGG16	16	138	Sequential	3x3 Convolutional layers
VGG19	19	144	Sequential	3x3 Convolutional layers
InceptionV3	48	23.8	Inception	Factorized convolutions, auxiliary classifiers
InceptionResNetV2	164	55.9	Inception + Residual	Combines Inception and Residual connections
DenseNet201	201	20.2	Densely Connected	Each layer connected to every other layer
EfficientNetB0	237	5.3	Compound Scaling	Scales depth, width, and resolution uniformly
NASNetLarge	88	84	Neural Architecture Search	Discovered via reinforcement learning

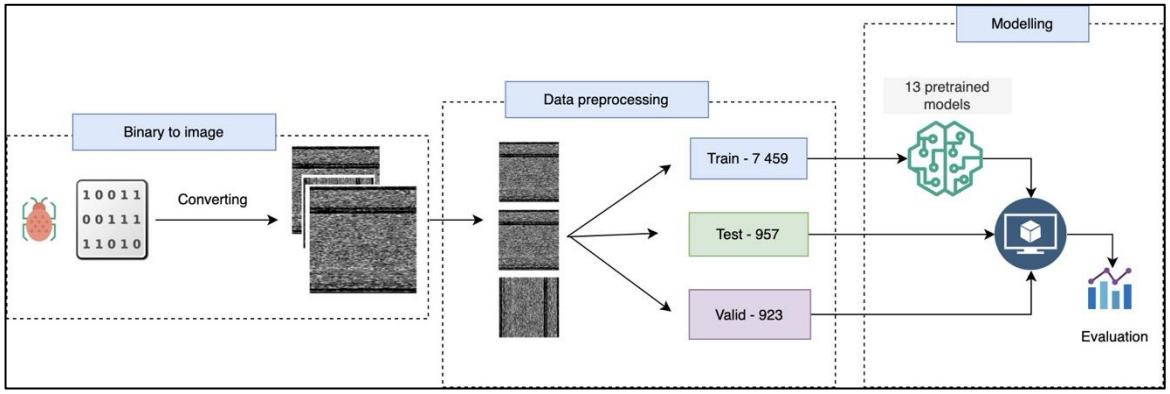


Fig. 1: Workflow of proposed method

Dataset description. The dataset used in this study is the Malimg Dataset [23], a comprehensive collection of malware images that has become a benchmark for malware detection and classification research. This dataset was created by converting malware binaries into grayscale images, a process that enables the application of image-based techniques for malware analysis. The Malimg Dataset includes various malware families, providing a diverse set of samples for training and evaluating machine learning models. The Malimg Dataset is organized into different directories, each representing a specific malware family. Within each directory are multiple grayscale images, each corresponding to a different instance of malware. These images are generated by interpreting the binary content of the malware files as pixel values, visually representing the malware's structure.

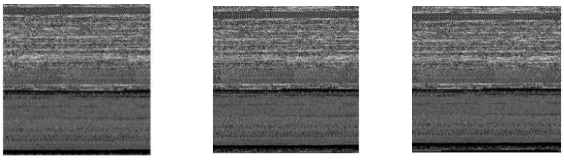
Each grayscale image is created by reading the binary content of a malware file and interpreting it as a sequence of pixel values. The image's width is fixed, and the length of the binary file determines the height. This approach transforms the binary data into a two-dimensional array, scaled to fit within the image dimensions. The resulting images highlight various structural patterns and features of the malware binaries, making them suitable for CNNs analysis. A list of some joint malware families in the dataset, along with the number of samples in each class, is shown in Table 2.

Table 2: Distribution of Samples in Malware Families

Malware Family	Number of Samples	Percentage (%)
Allaple.L	2,949	14.61
Allaple.A	2,949	14.61
C2Lop.gen!G	2,930	14.51
Fakerean	1,985	9.84
C2Lop.P	1,466	7.26
Lolyda.AA 3	1,239	6.13
Lolyda.AA 2	1,230	6.09
Lolyda.AA 1	1,133	5.61
Swizzor.gen!I	1,325	6.57
Swizzor.gen!E	1,320	6.55
Yuner.A	800	3.96
Instantaccess	431	2.13
VB.AT	439	2.18
Dontovo.A	162	0.80
Autorun.K	106	0.53
Rbot!gen	158	0.78
Alueron.gen!J	198	0.98
Malex.gen!J	136	0.67
Lolyda.AT	159	0.79
Adialer.C	125	0.62
Wintrim.BX	97	0.48
Dialplatform.B	177	0.88
Skintrim.N	80	0.40

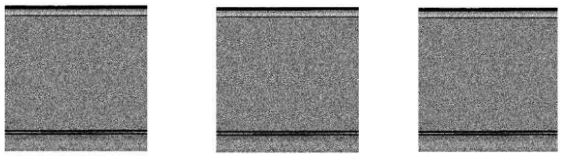
Figures 2 and 3 display examples of malware classes from the Malimg dataset: Instantaccess and Adialer. C. Each figure contains three samples from their respective classes. As observed, the samples within each class are pretty similar to one another, indicating a consistent pattern or structure characteristic of the class. However, when comparing samples across different classes, such as Instantaccess and Adialer. C, there are noticeable differences in their visual features, highlighting the distinctiveness between different malware classes.

Converting binaries to images. Generating images from malware binaries is a crucial step in applying image-based techniques for malware detection. This innovative approach involves converting the raw binary data of malware files into visual representations, allowing machine learning models, particularly convolutional neural networks (CNNs), to analyze and classify malware based on visual patterns and structures. The first step in this process is reading the binary content of each malware file. The binary file, essentially a sequence of bytes, is read into a buffer containing the raw data that represents the malware's code and structure. Once the binary content is read, it is reshaped into a two-dimensional array to form an image by setting a fixed width, while the height is determined by the total number of bytes in the file. This reshaping process transforms the linear sequence of bytes into a grid format, which can be visualized as a grayscale image. Each byte value, ranging from 0 to 255, corresponds to a pixel intensity, creating a visual representation that highlights the structural and content-based characteristics of the malware. This process is illustrated in Figure 4.



a) b) c)

Figure 2. Adiler C class samples



a) b) c)

Fig. 3. Instantaccess class samples

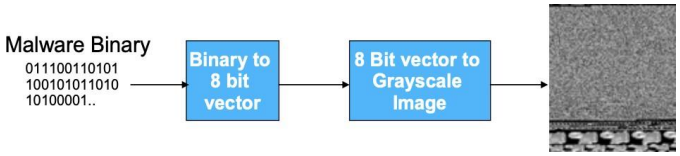


Fig. 4: Converting malware binary to image [23]

After the initial conversion, the images undergo further processing to ensure they are suitable for analysis by CNNs. This preprocessing includes resizing the images to a uniform dimension, typically 224x224 pixels, standardizing the input size for the neural network, and facilitating efficient training. Additionally, the pixel values are normalized to a range of [0, 1], an important step that scales the pixel values to a range optimal for neural network processing, helping to improve the model's convergence during training. The transformation of malware binaries into images reveals distinctive patterns indicative of different malware families. By analyzing these visual patterns, CNNs can learn to differentiate between various types of malware and benign software, leveraging the rich feature extraction capabilities of deep learning. This novel method of converting raw binary data into analyzable images provides a powerful tool for enhancing malware detection through advanced machine learning models.

Evaluation metrics. Multiple evaluation metrics were utilized to assess the performance of the machine learning algorithms. Given the multiclass nature of the dataset, these metrics were computed individually for each class. An aggregate metric was then derived by combining the individual class metrics to assess the model's performance holistically

Accuracy: As defined in Equation 1, Accuracy represents the ratio of correctly classified instances to the total instances in the dataset. It provides a general indication of the algorithms' effectiveness in accurately classifying Android malware.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FN + FP)} \quad (1)$$

Precision: As shown in Equation 2, Precision is the ratio of true positive predictions to all positive predictions. It measures the algorithms' ability to correctly identify malware instances without incorrectly classifying benign applications as malware.

$$Precision = \frac{TP}{(TP + FP)} \quad (2)$$

Recall (Sensitivity): Recall, as detailed in Equation 3, is the ratio of true positive predictions to all actual positive instances. It evaluates the algorithms' capability to detect all malware instances without missing any.

$$Recall = \frac{TP}{(TP + FN)} \quad (3)$$

F1-score: The F1-score, as illustrated in Equation 4, is the harmonic mean of precision and recall. It offers a balanced metric that considers both precision and recall, which is particularly beneficial in scenarios involving imbalanced datasets with varying numbers of malware and benign samples.

$$F1 - score = \frac{2 * precision * recall}{(precision + recall)} \quad (4)$$

In the formulas, TP (True Positives) represents the count of correctly predicted malware instances. TN (True Negatives) indicates the instances accurately predicted as benign. FP (False Positives) denotes the count of benign cases incorrectly classified as malware. FN (False Negatives) signifies the number of malware instances mistakenly classified as benign.

Experiments and results

Experimental setup. The experiments for this study were conducted on the Kaggle platform, leveraging its robust computational resources to facilitate efficient model training and evaluation. The environment provided several configurations, including CPU, GPU, and TPU instances. For CPU and GPU notebook sessions, the execution time was capped at 12 hours, while TPU notebook sessions had a maximum execution time of 9 hours. The platform offered 20 gigabytes of auto-saved disk space and additional scratchpad disk space that was not preserved outside the current session.

The CPU setup included 4 CPU cores and 30 gigabytes of RAM, providing a balanced configuration for initial data preprocessing and model training. The P100 GPU configuration featured 1 Nvidia Tesla P100 GPU, 4 CPU cores, and 29 gigabytes of RAM, offering enhanced computational power for more intensive deep learning tasks. For even more excellent performance, the T4 x2 GPU setup included 2 Nvidia Tesla T4 GPUs alongside 4 CPU cores and 29 gigabytes of RAM, facilitating faster model training and improved handling of larger datasets. The TPU 1VM configuration was the most powerful, with 96 CPU cores and 330 gigabytes of RAM, designed to handle extensive deep learning workloads with high parallelism and efficiency. This diverse range of computational resources ensured that all experiments were conducted efficiently, with adequate processing power, to achieve reliable and accurate results.

Data preprocessing. This study's data preprocessing pipeline was meticulously designed to prepare the Malimg dataset for practical training of deep learning models. The dataset, comprising malware binaries converted into grayscale images, required several key preprocessing steps to ensure consistency and optimize model performance.

Initially, the dataset was divided into training, validation, and test sets. The data and corresponding labels were read from their respective directories for each set. The images were resized to a uniform dimension of 224x224 pixels using OpenCV. This resizing ensured that all images had a consistent input size, facilitating efficient training across

different neural network architectures. After resizing, the images were converted into NumPy arrays and organized into their respective datasets (training, validation, and test). The data within each set was shuffled to ensure randomness and avoid any potential bias. The labels were then encoded using the LabelEncoder from scikit-learn, transforming the categorical labels into numerical values suitable for the model. These numerical labels were subsequently converted into one-hot encoded vectors using the to_categorical function from Keras, which is crucial for the categorical cross-entropy loss function used during training. The final preprocessed datasets were then used for model training and evaluation.

Result of experiments. Table 3 and 4 comprehensively compare the performance metrics for various pre-trained models fine-tuned using two different optimizers: Adam and SGD, respectively. The metrics evaluated include accuracy, precision, recall, and F1-score. The results show a consistent trend where the Adam optimizer generally outperforms the SGD optimizer across most models and metrics.

Table 3: Performance comparison of various models using Adam optimizer

Models	**Accuracy**	**Precision**	**Recall**	**F1-score**
ResNet50	***98.5***	***98.6***	***98.5***	***98.5***
ResNet101	97	97	98	97
ResNet152	98	98	98	98
Xception	***98***	***98.7***	***98.5***	***98.5***
MobileNet	***98.8***	***98.9***	***98.9***	***98.9***
MobileNetV2	***98.7***	***98.9***	***98.7***	***98.6***
VGG16	82	83	82	81
VGG19	86	86	86	85
InceptionV3	97	95	97	96
InceptionResNetV2	97	96	97	96
DenseNet201	***99.0***	***99.0***	***99.0***	***99.0***
EfficientNetB0	98	98	98	98
NASNetLarge	98	98	98	98

Table 4: Performance comparison of various models using SGD optimizer

Models	**Accuracy**	**Precision**	**Recall**	**F1-score**
ResNet50	96	95	96.5	96
ResNet101	96	94	96	95
ResNet152	97	96	97	96
Xception	93	90	93	91
MobileNet	97	96	97	96
MobileNetV2	97	96	97	96
VGG16	76	76	77	74
VGG19	12	9	12	8
InceptionV3	96	95	96	96
InceptionResNetV2	95	95	95	94

DenseNet201	97	96	97	96
EfficientNetB0	94	93	94	93
NASNetLarge	92	91	93	92

DenseNet201, Xception, MobileNet, MobileNetV2, and ResNet50 are among the top-performing models, achieving near-perfect scores across all metrics. DenseNet201 stands out with an impressive 99.0% accuracy, precision, recall, and F1-score, indicating its exceptional capability in accurately classifying malware. Similarly, MobileNet and MobileNetV2 exhibit excellent performance with accuracy and F1-scores around 98.8% and 98.7%, respectively. These models provide high accuracy and maintain a strong balance between precision and recall, ensuring that both true positive and true negative rates are maximized.

Top-performing models' training and validation accuracy and loss lines by epoch are shown in Figures 5, 6, 7, 8, and 9 for ResNet50, DenseNet201, Xception, MobileNetV2, and MobileNet, respectively. As observed, only the MobileNet model exhibits training and validation lines that are closer together, indicating its robustness and consistency in performance across different epochs. This closer alignment suggests that MobileNet generalizes better and is less prone to overfitting than the other models, making it a highly practical choice for malware detection using image-based deep learning techniques.

Table 5 illustrates the effectiveness of various malware detection methods applied to the Malimg dataset. Different studies have reported F1-scores ranging from approximately 70% to 98%. The proposed approach using MobileNet achieves a remarkable F1-score of 98.9%, demonstrating its superior performance in comparison to the other methods listed.

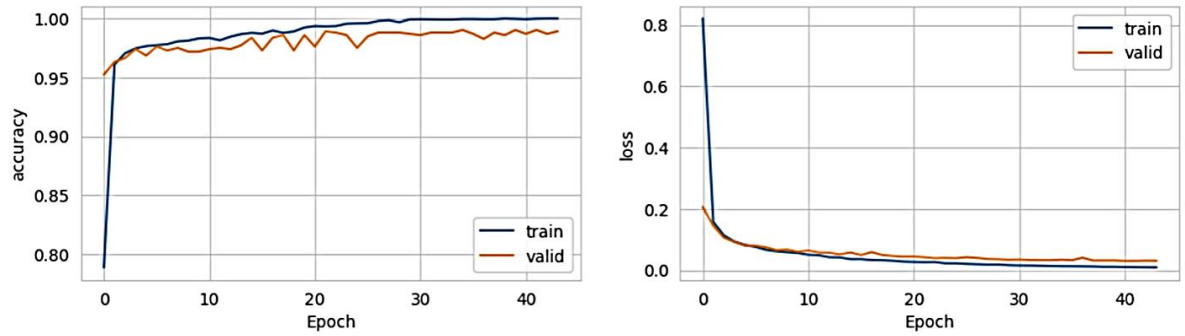


Fig. 5: ResNet50

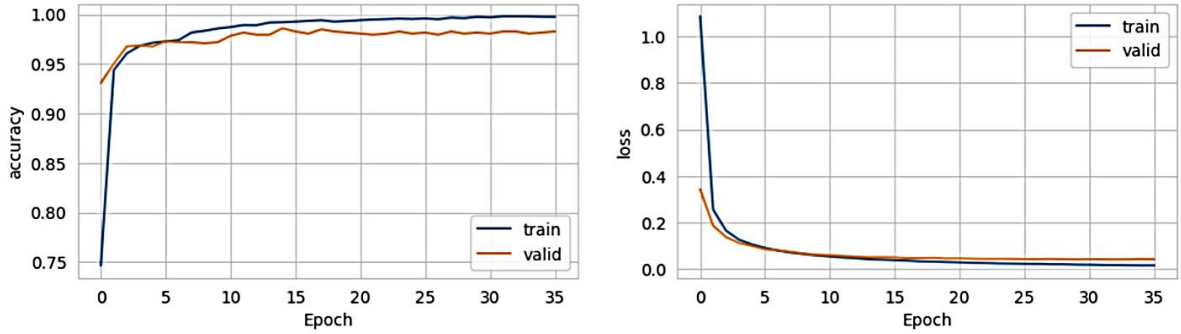


Fig. 6: DenseNet201

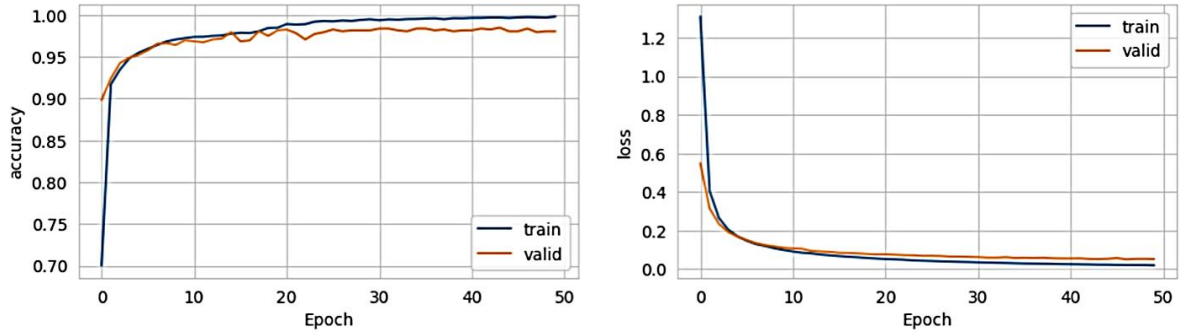


Fig. 7: Xception

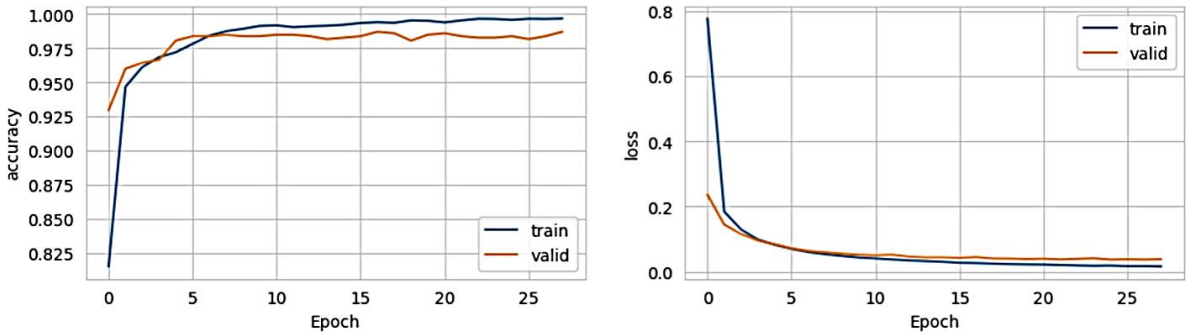


Fig. 8: MobileNetV2

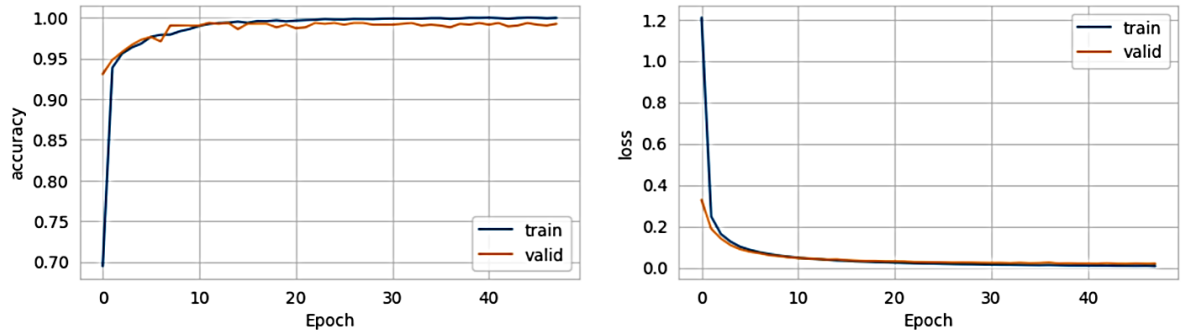


Fig. 9: MobileNet

Table 5: Comparison of other authors' works with this work using F1-score

Author	Year	Suggested method	F1-score (%)
[24]	2023	Segmentation-based fractal texture analysis and KNN	70.14
[25]	2023	Custom CNN model	98, but overfitting after 5th epoch
[26]	2023	Butterfly construction-based vision transformer (B_ViT) model	74.29
[27]	2023	Attention-based Cross-modal CNN	97.6
[28]	2024	EfficientNetB0	97
This work	***2024***	***MobileNet***	***98.9***

Conclusions

Summary. The application of artificial intelligence offers significant potential for solving complex problems across various domains, including healthcare, finance, and autonomous systems [29-32]. This study applied an image-based analysis approach for malware detection utilizing deep learning techniques on the Malimg dataset. Using malware binaries represented as grayscale images, advanced convolutional neural network architectures such as VGG16, ResNet50, and MobileNet were leveraged to extract high-level features and classify various malware families. Experiments on Kaggle's computational platform included hyperparameter tuning to optimize model performance. This comprehensive approach demonstrated the effectiveness of these models in accurately distinguishing between different types of malware.

The trained models on the Malimg dataset demonstrated high effectiveness in malware detection through image-based analysis. Models such as DenseNet201, MobileNet, and ResNet50 achieved notable accuracy, precision, recall, and F1 scores, indicating their robustness in classifying various malware families. Hyperparameter tuning further optimized their performance, showcasing the models' ability to generalize well across the dataset. When comparing optimizers, models trained with the Adam optimizer consistently outperformed those trained with the SGD optimizer.

The training and validation accuracy analysis over epochs for various models revealed insightful trends, particularly highlighting MobileNet's robustness. Among the models, MobileNet showed the closest alignment between training and validation accuracy lines, suggesting better generalization and minimal overfitting. This consistency indicates that MobileNet is particularly effective in maintaining high performance across different data splits, making it a reliable choice for malware detection tasks. The comparison across epochs for models like DenseNet201, ResNet50, and Xception also demonstrated strong performance. Still, the slight divergence between training and validation curves suggested a higher tendency towards overfitting compared to MobileNet. These findings emphasize the importance of monitoring model performance over epochs to ensure optimal training and robust real-world application.

Limitations. While this study demonstrates the efficacy of using deep learning techniques on the Malimg dataset for malware detection, several limitations exist. The image-based analysis approach, although practical, can be computationally intensive, requiring significant resources for training and inference, which may not be feasible in all practical scenarios. The models trained in this study must also be evaluated for their robustness against adversarial attacks, as deep learning models can be susceptible to such

threats. Moreover, real-time deployment in production environments poses latency and computational overhead challenges, which were not fully addressed in this research.

Future works. Future research can build upon the findings of this study by exploring several avenues. Enhanced preprocessing techniques, such as image augmentation and feature engineering, could be investigated to improve the robustness and accuracy of the models. Additionally, combining image-based analysis with other data modalities, such as network traffic or behavioral analysis, may provide a more comprehensive malware detection framework.

Developing and optimizing models for real-time malware detection in production environments would be a valuable extension of this work, involving the reduction of computational overhead and latency associated with model inference. Furthermore, exploring the impact of adversarial attacks on the proposed models and developing techniques to enhance their robustness against such threats is crucial for practical deployment. Lastly, examining the scalability of the proposed approach for large-scale deployment in enterprise environments, including the integration with existing security infrastructure and workflows, is another critical area for future research. Future research can further advance the field of malware detection, contributing to more secure and resilient computing environments.

Declarations

The authors did not receive support from any organization for the submitted work. During the preparation of this work, the author used "ChatGPT" to language editing and refinement. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

References

1. Awan M.J., Masood O.A., Mohammed M.A., et al. Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention // Electronics. 2021. Vol. 10(19). Article ID 2444. https://doi.org/10.3390/electronics10192444
2. Baghirov E. Malware detection based on opcode frequency // Journal of Problems of Information Technology. 2023. No. 1. P. 3–7.
3. Baghirov E. Evaluating the performance of different machine learning algorithms for Android malware detection // Proceedings of the 5th International Conference on Problems of Cybernetics and Informatics. 2023. August 28–30.
4. Yajamanam S., Selvin V.R.S., Di Troia F., et al. Deep Learning versus Gist Descriptors for Image-based Malware Classification // Proceedings of the 4th International Conference on Information Systems Security and Privacy - ForSE. SciTePress, 2018. P. 553–561. https://doi.org/10.5220/0006685805530561
5. Neurosci. Volume 2022. Article ID 7671967, 17 pages. Hindawi. https://doi.org/10.1155/2022/7671967
6. Kim J., Ban Y., Ko E., et al. MAPAS: A practical deep learning-based Android malware detection system // International Journal of Information Security. 2022. Vol. 21. P. 725–738. https://doi.org/10.1007/s10207-022-00579-6
7. Mimura M., Ito R. Applying NLP techniques to malware detection in a practical environment // International Journal of Information Security. 2022. Vol. 21. P. 279–291. https://doi.org/10.1007/s10207-021-00553-8
8. Khan I., Kwon Y.W. Multi-class Malware Detection via Deep Graph Convolutional Networks Using TF-IDF-Based Attributed Call Graphs // In: Kim H., Youn J. (eds). Inf. Secur. Appl. WISA 2023. Lecture Notes in Computer Science. Vol. 14402. Springer, 2023. https://doi.org/10.1007/978-981-99-8024-6_15
9. Daoudi N., Samhi J., Kabore A.K., et al. DEXRAY: A Simple, yet Effective Deep Learning Approach to Android Malware Detection Based on Image Representation of Bytecode // MLHat 2021. Communications

in Computer and Information Science. Vol. 1482. Springer, 2021. https://doi.org/10.1007/978-3-030-87839-9_4
10. Zhao Z., Zhao D., Yang S., et al. Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model // Security and Communication Networks. 2023. Article ID 6390023. https://doi.org/10.1155/2023/6390023
11. Kumar S., Panda K. SDIF-CNN: Stacking deep image features using fine-tuned convolutional neural network models for real-world malware detection and classification // Applied Soft Computing. 2023. Vol. 146. Article ID 110676. https://doi.org/10.1016/j.asoc.2023.110676
12. Pachhala N., Jothilakshmi S., Battula B.P., et al. Enhanced malware family classification via image-based analysis utilizing a balance-augmented VGG16 model // Traitement du Signal. 2023. Vol. 40(5). P. 2169–2178. https://doi.org/10.18280/ts.400534
13. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // arXiv preprint arXiv:1409.1556. 2015.
14. He K., Zhang X., Ren S., et al. Deep Residual Learning for Image Recognition // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 770–778.
15. Hai T.H., Van Thieu V., Duong T.T., et al. A Proposed New Endpoint Detection and Response With Image-Based Malware Detection System // IEEE Access. 2023. Vol. 11. P. 122859–122875. https://doi.org/10.1109/ACCESS.2023.3329112
16. Sandler M., Howard A., Zhu M., et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. P. 4510–4520.
17. Howard A.G., Zhu M., Chen B., et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications // arXiv preprint arXiv:1704.04861. 2017.
18. Huang G., Liu Z., Van Der Maaten L., et al. Densely Connected Convolutional Networks // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 4700–4708.
19. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 1251–1258.
20. Szegedy C., Vanhoucke V., Ioffe S., et al. Rethinking the Inception Architecture for Computer Vision // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 2818–2826.
21. Tan M., Le Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // International Conference on Machine Learning (ICML). 2019. P. 6105–6114.
22. Zoph B., Vasudevan V., Shlens J., et al. Learning Transferable Architectures for Scalable Image Recognition // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. P. 8697–8710.
23. Nataraj L., Karthikeyan S., Jacob G., et al. Malware images: Visualization and automatic classification // Proceedings of the 8th International Symposium on Visualization for Cyber Security. 2011. P. 1–7.
24. Ahmed I.T., Hammad B.T., Jamil N. A Comparative Performance Analysis of Malware Detection Algorithms Based on Various Texture Features and Classifiers // IEEE Access. 2024. Vol. 12. P. 11500–11519. https://doi.org/10.1109/ACCESS.2024.3354959
25. Jabra M.B., Cheikhrouhou O., Atitallah N., et al. Malware Detection Using Deep Learning and CNN Models // Proceedings of the 2023 International Conference on Cyberworlds (CW). 2023. P. 432–439. https://doi.org/10.1109/CW58918.2023.00073
26. Hai T.H., Van Thieu V., Duong T.T., et al. A Proposed New Endpoint Detection and Response With Image-Based Malware Detection System // IEEE Access. 2023. Vol. 11. P. 122859–122875. https://doi.org/10.1109/ACCESS.2023.3329112
27. Kim J., Paik J.Y., Cho E.S., et al. Attention-Based Cross-Modal CNN Using Non-Disassembled Files for Malware Classification // IEEE Access. 2023. Vol. 11. P. 22889–22903. https://doi.org/10.1109/ACCESS.2023.3253770
28. Habib F., Shirazi S.H., Aurangzeb K., et al. Deep Neural Networks for Enhanced Security: Detecting Metamorphic Malware in IoT Devices // IEEE Access. 2024. Vol. 12. P. 48570–48582. https://doi.org/10.1109/ACCESS.2024.3383831
29. Потреба, Е.Ю. Анализ Методов И Средств Предотвращения Утечек Конфиденциальных Данных // Е.Ю. Потреба, Н.Е. Губенко // Проблемы искусственного интеллекта. - 2023. № 3 (30). https://doi.org/10.34757/2413-7383.2023.30.3.005
30. Kharlamov, A. Semantic Text Analysis Using Artificial Neural Networks Based On Neural-Like Elements With Temporal Signal Summation // A. Kharlamov, E. Samaev, D. Kuznetsov и др. // Проблемы искусственного интеллекта. - 2023. № 3 (30). https://doi.org/10.34757/2413-7383.2023.30.3.001
31. Pikalyov, Ya.S. About Neural Architectures Of Feature Extraction For The Problem Of Object Recognition On Devices With Limited Computing Power // Ya.S. Pikalyov, T.V. Yermolenko // Проблемы искусственного интеллекта. - 2023. № 3 (30). https://doi.org/10.34757/2413-7383.2023.30.3.004

32. Nikitina, A.A. Detection Of Objects On The Ground By Intelligent Robots In A Rapidly Changing Environment // A.A. Nikitina, S.I. Ulanov // Проблемы искусственного интеллекта. - 2023. № 3 (30). - https://doi.org/10.34757/2413-7383.2023.30.3.003

RESUME

Y. Imamverdiyev, E. Baghirov, I.J. Chukwu
Image-based Deep Learning Method for Effective Malware Detection

The detection of malware using traditional methods is becoming increasingly challenging due to the growing sophistication and complexity of malicious software. The article proposes an innovative method for malware detection based on image analysis. This approach converts malware binaries into grayscale images and applies advanced deep learning techniques for their classification. A total of 13 pre-trained convolutional neural network (CNN) architectures, including DenseNet201 and MobileNet, are utilized for feature extraction and classification. The experiments were conducted on the Malimg dataset, and the models were optimized through hyperparameter tuning.

The proposed method is notable for its efficiency and resilience, as it does not require executing or disassembling malware samples, unlike traditional dynamic and static analysis methods. The hierarchical structure of the approach ensures robust detection, where high-level features of malware are extracted and analyzed for classification. DenseNet201 and MobileNet demonstrate exceptional performance, achieving high accuracy, precision, recall, and F1-scores across all metrics.

The method's core contribution is its ability to adaptively enhance malware detection using image-based deep learning techniques, emphasizing safety and computational efficiency. This approach has significant implications for modern cybersecurity, particularly in addressing the challenges posed by sophisticated and evasive malware, thus providing a novel solution for advanced threat detection and prevention.

РЕЗЮМЕ

Я. Имамвердиева, Э. Багиров, И.Дж. Чукву
Метод глубокого обучения на основе изображений для эффективного обнаружения вредоносного ПО

Обнаружение вредоносного программного обеспечения с использованием традиционных методов становится все более сложной задачей из-за растущей изощренности и сложности вредоносного ПО. В статье предлагается инновационный метод обнаружения вредоносного ПО, основанный на анализе изображений. Этот подход преобразует бинарные файлы вредоносного ПО в изображения в оттенках серого и применяет современные методы глубокого обучения для их классификации. Для извлечения признаков и классификации использовано 13 предобученных архитектур сверточных нейронных сетей (CNN), включая DenseNet201 и MobileNet. Эксперименты были проведены на наборе данных Malimg, а модели были оптимизированы с помощью настройки гиперпараметров.

Предложенный метод отличается эффективностью и устойчивостью, так как не требует выполнения или дизассемблирования образцов вредоносного ПО, в отличие от традиционных методов динамического и статического анализа. Иерархическая структура подхода обеспечивает надежное обнаружение, при котором высокоуровневые признаки вредоносного ПО извлекаются и анализируются для классификации. DenseNet201 и MobileNet демонстрируют исключительную производительность, достигая высокой точности, полноты, точности и F1-метрик по всем показателям.

Основной вклад метода заключается в его способности адаптивно улучшать процесс обнаружения вредоносного ПО с использованием методов глубокого обучения на основе изображений, с акцентом на безопасность и вычислительную эффективность. Этот подход имеет значительные последствия для современной кибербезопасности, особенно при решении задач, связанных с изощренными и уклоняющимися вредоносными программами, предоставляя новое решение для передового обнаружения и предотвращения угроз.

Yadigar Imamverdiyev – **Associate professor,** Azerbaijan Technical University, H. Javid, 25, Baku, Azerbaijan, phone: (+994 50) 540-74 64, yadigar.imamverdiyev@aztu.edu.az.
Area of scientific interest: Information security, biometric technologies, e-government security, artificial intelligence in security, cryptographic systems, social network analysis, distributed computing, mathematical logic.

Elshan Baghirov – PhD Candidate, Institute of Information Technology, Ministry of Science and Education of the Republic of Azerbaijan, B. Vahabzade, 9A, Baku, Azerbaijan, phone: (+994 51) 444-19-33, elsenbagirov1995@gmail.com.
Area of scientific interest: machine learning, cybersecurity, malware detection.

Ikechukwu John Chukwu – Researcher, Kadir Has University, Istanbul, Türkiye, and Ss. Cyril and Methodius University in Skopje (UKIM), North Macedonia, ikechukwu@khas.edu.tr
Area of scientific interest: deep learning, image-based analysis, software security.

The article was submitted to the editorial office on 30.11.2024