Malware Detection Using Convolutional Neural Networks: A Visual Approach

Hamza Amin
GIKI
mh4070685@gmail.com

Ali Raza
GIKI
ali.dot.raza@gmail.com

***Abstract*—In the rapidly evolving landscape of cybersecurity, malware detection and classification are critical challenges for protecting systems and networks. Traditional methods, such as signature-based detection, often fail to adapt to novel and obfuscated malware. This paper presents a novel approach leveraging Convolutional Neural Networks (CNNs) to classify malware samples represented as grayscale images. By converting malware binaries into visual representations, the proposed methodology utilizes a combination of handcrafted features and deep learning-based feature extraction to capture intricate structural and behavioral patterns. Additionally, a feature fusion strategy integrates these representations to enhance classification performance. The proposed model demonstrates superior accuracy and robustness in detecting and classifying malware, as evidenced by comprehensive experimental evaluations. This approach holds potential for significant advancements in automated malware detection.**

Keywords— malware detection, convolutional neural networks, grayscale image conversion, feature fusion, handcrafted features, deep learning.

I. Introduction

Global cybersecurity experts face a difficult task as a result of the exponential rise in cyberthreats in recent years. A crucial component of many cyberattacks, malware is always becoming more complicated and utilizing advanced strategies like polymorphism and obfuscation to get past conventional detection methods. Although they work well against known malware, traditional signature-based detection systems frequently miss zero-day threats and new variations, placing vital systems at risk. Because of these constraints, it is necessary to investigate sophisticated and flexible malware detection techniques that can successfully address the constantly shifting threat landscape.

Worldwide network safety specialists face a troublesome errand because of the remarkable ascent in cyberthreats lately. A significant part of numerous cyberattacks, malware is continuously turning out to be more confounded and using progressed methodologies like polymorphism and confusion to move beyond regular identification strategies. In spite of the fact that they function admirably against known malware, customary mark based discovery frameworks regularly miss zero-day dangers and new varieties, setting imperative frameworks in danger. Due to these requirements, it is important to examine modern and adaptable malware identification procedures that can effectively address the continually moving danger scene.

The exploration question that supports this review is: *Can an element combination approach joining hand tailored and profound learning-based highlights work on the precision and strength of malware order utilizing visual representations?* To resolve this inquiry, this study presents an extensive strategy that incorporates hand tailored highlights, Gray Level Co-occurrence Matrix (GLCM), Local Binary Patterns (LBP), and Global Image Descriptors (GIST), with profound highlights removed utilizing CNNs. The combination of these component types is intended to bridle the qualities of the two methodologies — catching point by point textural designs through hand tailored descriptors while utilizing undeniable level reflections from CNNs.

The essential goals of this study are triple: 1. To change malware parallels into grayscale pictures and examine the visual examples that separate malware families. 2. To foster a CNN-based structure that integrates both handmade and profound learning-based include extraction procedures for improved order. 3. To assess the viability of the proposed approach utilizing genuine world malware datasets and give a near examination of various component extraction techniques.

This exploration additionally tries to lay out the pragmatic significance of the proposed technique by zeroing in on datasets that reflect genuine difficulties in malware identification. The incorporation of hand tailored highlights guarantees that basic low-level qualities are not ignored, while profound elements give a more extensive context oriented comprehension of the information. Together, these methodologies expect to make a vigorous recognition system that isn't just exact yet in addition computationally effective.

The extent of this paper stretches out past proposing a clever strategy; it likewise incorporates a thorough assessment of the proposed approach against existing procedures. By leading an exhaustive examination, this study contributes significant bits of knowledge into the transaction among hand tailored and profound learning-based highlights, featuring their separate assets and shortcomings in malware discovery.

II. Related Work

Authors in [8] carried out a picture-based approach to identify malicious patterns in code. They designed an arrangement of API calls to match RGB pictures. After preprocessing and preparing the data for the neural network, it was fed into a Convolutional Neural Network (CNN). They tackled

a two-class problem, i.e., identifying whether an application is benign or malicious. The authors ran the experiment for 100 epochs with a batch size of 32. A disassembling process was used to separate API calls. Authors in [**?**] considered a dataset of 144 Android permissions. They utilized tools such as a manifest parser and a small disassembler for parsing and decompilation of an APK. Further, they extracted the mentioned permissions from the disassembled manifest file and transformed it into 12×12 permission vectors as an image. Their dataset contains a total of 2500 Android applications, of which 2000 applications were malware samples and 500 applications were benign samples. Furthermore, a deep learning model was applied to detect and classify malware samples.

Authors in [10] first disassembled the APK file and extracted only the dex bytecode from the file. They converted the dex bytecode into RGB picture format and fed it into a Convolutional Neural Network (CNN) for automatic feature extraction and training. Authors in [11] implemented their Android malware detection approach in two stages. In the first stage, they extracted the dex bytecode from the APK file and converted it into RGB pictures. In the second stage, the pictures were used to train the Convolutional Neural Network (CNN).

Authors in [12] used the Convolutional Neural Network (CNN) for detection of Android malware. They utilized the Rectified Linear Unit (ReLU) activation function as it overcomes the vanishing gradient problem and shows better assembly performance. Authors in [13] implemented a multimodal deep learning approach for Android malware detection. They utilized publicly available datasets Android and Knowledge Discovery in Databases (KDD) to train and evaluate the proposed model. Authors in [14] used semantic features for the classification of malware families. The classifiers such as Support Vector Machines, K-Nearest Neighbor, Random Forest, and Naive Bayes were used in the experimentation. The results with Support Vector Machines achieved the highest accuracy of 92.7

Authors in [15] transformed the Dalvik executable code into a layered bytecode graph. Further, a Convolutional Neural Network (CNN) was used for training and classification tasks. Convolutional Neural Networks can automatically learn the features from the bytecode files to recognize malware.

Different research areas and trends in the Android security space were focused on by authors involving dormant semantic analysis approach in [16], [17]. Authors in [18] discussed the disturbing challenges in the field of Android security. Authors in [19] implemented the TensorFlow models such as GoogleNet and ResNet for malware detection. In their work, ResNet turned out to be more accurate but consumed a lot of time. Authors in [20] proposed the image texture-based approach to perform the analysis on the code. They combined the image texture features and API calls to train the Deep Belief Network (DBN). DBN is stacked with Restricted Boltzmann Machines (RBM) and Back Propagation (BP). Authors also compared their proposed model with shallow machine learning models such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and shallow feed-forward network ANN (Artificial Neural Network) and found that the proposed DBN model was more accurate.

Table 1 includes a summary of related literature. The literature study revealed that approaches to analyzing malware include static analysis and dynamic analysis or perhaps a combination of both. The static analysis primarily focuses on disassembling the code, followed by manual analysis to search for malicious patterns in the code. On the other hand, dynamic analysis executes the code in a virtual environment and analyzes its execution trace to observe the malicious behavior of an application. The static analysis is helpful in tracing unique and full execution paths; thus, it provides complete code coverage but ultimately it suffers from code obfuscation. The code must be deobfuscated first to perform static analysis. The issues of resilient obfuscation hinder the analysis. Dynamic analysis is more efficient and does not require the executable to be unloaded or deobfuscated. The suspicious application is monitored in a controlled environment. This process is time and resource-consuming. It also raises portability issues. Moreover, some malicious behavior may go unnoticed because the environment does not satisfy the triggering conditions. Furthermore, malware authors use automation technology to generate a huge number of new malware variants, thus posing a big challenge to malware investigators. The current state of the art demands the integration of existing crude methods with beneficial strategies to achieve an effective solution. Beneficial strategies, such as representation-based analysis, ought to be utilized to supplement the arrangement of quickly developing Android malware families. It is demonstrated to be viable in deciding unusual present-day noxious way of behaving or security weaknesses. Sending a representation-based strategy, a malware variation can be envisioned as a picture. A picture can catch even little changes. In this paper, the representation-based strategy upheld with highlight combination technique is proposed to diminish the impact of confusion by changing the malware's non-instinctive highlights into unique finger impression pictures followed by the characterization of Android malware families. The accompanying segment makes sense of the received philosophy and receives a contextual investigation.

III. Dataset Preparation

In order to facilitate efficient visual-based analysis, the dataset used in this work includes samples from both malware and benign categories that have undergone careful processing. Every sample is transformed into a grayscale image, with the binary data of the associated file represented by the pixel intensity values. The virus's subtle structural and behavioral characteristics are captured by this transformation, making it possible to spot distinctive patterns that can be used to categorize the malware. The dataset's visual representation of files makes it possible to use sophisticated image processing methods and machine learning algorithms—more especially, Convolutional Neural Networks (CNNs)—to differentiate between dangerous and benign software.

A. Scatter Plot of Image Dimensions

To analyze the dataset, a bar plot of the data distribution was generated, as shown in Figure 1. The plot features the count of benign and malware tests inside the dataset. The benign class has a count of roughly 2600, while the malware class has a count of around 3000. This distribution gives bits of knowledge into the equilibrium of the dataset, which is a basic calculation of model preparation and execution. A fair dataset guarantees that the model doesn't become one-sided towards one class, prompting more precise and dependable expectations. This highlights the significance of having a very dispersed dataset for compelling preparation of the grouping model.

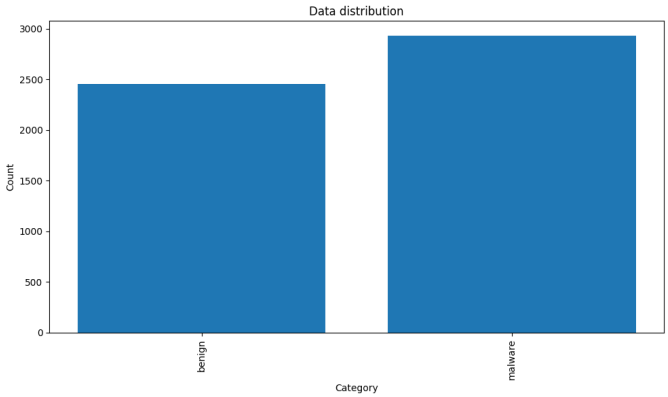

Fig. 1. Data distribution of the images.

B. Sample Images from Dataset

Figure 2 provides examples of grayscale images from the benign and malware categories. These pictures clearly delineate the unmistakable examples and surfaces innate in every class. Benign pictures for the most part display smooth and dull examples, characteristic of coordinated document structures. Conversely, malware pictures frequently show sporadic and tumultuous examples, mirroring the muddled and peculiar nature of pernicious code. These visual contrasts are vital to the grouping system, as they empower CNN to really learn and recognize trademark highlights of every classification.

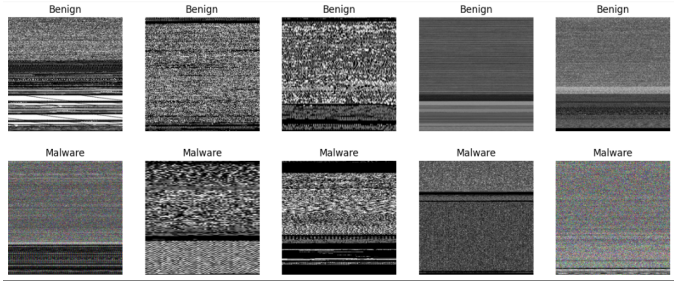

Fig. 2. Sample images: benign (top) and malware (bottom).

C. Preprocessing images

The dataset used for this project consists of grayscale images derived from both benign and malware samples. Each byte of a file is represented as a pixel value in these images, transforming the raw data into a format suitable for pattern recognition by CNNs. Figure 3 illustrates key preprocessing steps applied to our dataset images. First, images are resized to a standardized dimension, ensuring consistency for subsequent analysis and model training. Next, we enhance contrast to make subtle data variations more pronounced, aiding in the detection of intricate patterns. Additionally, edge detection, indicated by yellow lines, highlights significant intensity transitions, which are critical for identifying structural boundaries within the images. Finally, converting images to grayscale simplifies them by focusing solely on intensity values, eliminating color information that is not pertinent to our analysis. These preprocessing techniques collectively improve the dataset's quality and facilitate more effective model training and evaluation.

IV. Methodology

This section provides an in-depth explanation of the methodology utilized in our research. Two distinct approaches for malware detection are presented, each designed to address the challenges of malware classification in unique ways. The methodologies are described in detail, highlighting the tools, techniques, and processes used in their implementation.

A. Convolutional Neural Network (CNN)-Based Malware Detection

The first approach involves the development and training of a convolutional neural network (CNN) to classify malware samples based on their grayscale image representations. By transforming malware binaries into images, we leverage the spatial patterns in the data for classification.

1) Data Transformation and Preprocessing: Malware samples and benign software were converted into grayscale images, where each pixel corresponds to a byte value from the binary file. This conversion highlights structural patterns unique to each class. The images were resized to a uniform dimension of 224×224 pixels to ensure consistency during training.

The dataset was split into training, validation, and test sets. Data augmentation techniques, such as random rotations, flips, and zoom, were applied to mitigate overfitting and enhance generalization.

2) Model Architecture: The CNN model consists of multiple layers, including convolutional layers with ReLU activation functions, max-pooling layers for dimensionality reduction, and batch normalization layers to stabilize the training process. The final layers are fully connected, concluding with a softmax classifier for binary classification. Dropout regularization was employed to prevent overfitting.

The model was implemented using TensorFlow and optimized using the Adam optimizer with a learning rate of 0.001. The binary cross-entropy loss function was used to measure

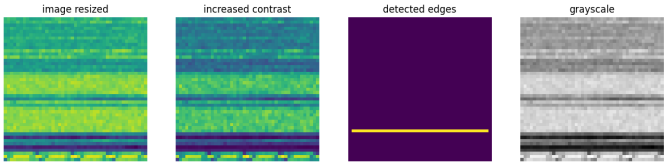

Fig. 3. Preprocessing of the image.

the model's performance. Figure 4 shows the first few layers of our CNN model.

3) Implementation and Tools: This approach utilized Python and TensorFlow for model development. Image data preprocessing and augmentation were performed using the ImageDataGenerator class from TensorFlow. The experiments were conducted on a high-performance GPU environment.

B. Handcrafted Features and Transfer Learning

The second approach combines handcrafted features with features extracted via transfer learning. Handcrafted feature descriptors provide domain-specific insights, while transfer learning captures high-level patterns in the data.

1) Handcrafted Feature Extraction: Features were extracted using the following techniques:

- **Gray Level Co-occurrence Matrix (GLCM):** Extracts texture features such as contrast and correlation. - **Local Binary Pattern (LBP):** Encodes local texture by thresholding pixel intensities. - **Global Image Descriptor (GIST):** Captures global spatial structures.

The features were computed for each grayscale image, normalized, and stored for subsequent analysis.

2) Transfer Learning with MobileNetV2: Features were extracted from the penultimate layer of the MobileNetV2 model, pre-trained on ImageNet. This provided a high-dimensional feature vector summarizing the image's global characteristics. The MobileNetV2 model was implemented using TensorFlow, with image resizing and preprocessing performed to align with the model's input requirements.

3) Feature Fusion and Classification: The handcrafted and MobileNetV2 features were concatenated to form a unified feature vector. A Random Forest classifier was trained on this combined feature set. This classifier, implemented using Scikit-learn, was optimized to handle the high dimensionality of the feature space.

4) Implementation Details: Python, Scikit-learn, and TensorFlow were the primary tools for this approach. The Random Forest classifier was configured with 100 estimators and trained using the concatenated feature vectors. Feature importance metrics were analyzed to evaluate the contributions of each feature type.

C. Summary of Tools and Environment

Both approaches were implemented using Python. TensorFlow facilitated deep learning model development and transfer learning, while Scikit-learn was employed for traditional machine learning methods. Experiments were executed on a GPU-accelerated environment, enabling efficient handling of computationally intensive tasks.

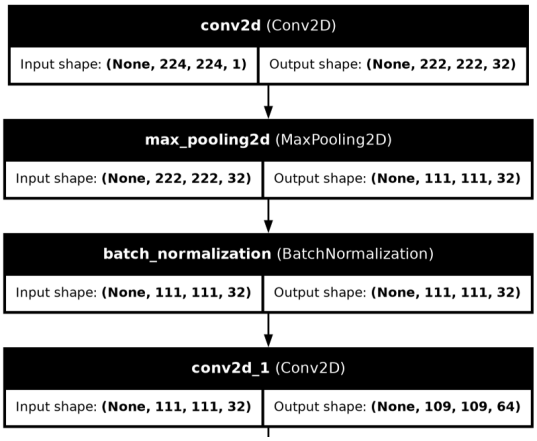

Fig. 4. CNN Model.

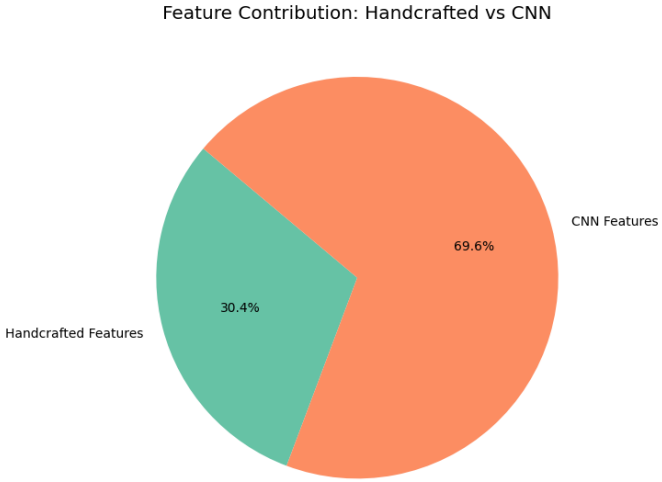

Fig. 5. Percentage of Handcrafted and CNN Feature Extraction.

D. Conclusion

The methodologies outlined above provide distinct yet complementary approaches to malware detection. The CNN-based method leverages end-to-end learning for automated feature extraction, while the second approach integrates domain-specific knowledge with advanced transfer learning techniques, offering robust and interpretable solutions.

V. Results and Discussion

A. Performance Comparison

The performance of the proposed methodologies was evaluated across multiple metrics, including accuracy, precision, recall, and F1-score. A detailed comparison of the feature extraction methods is presented in Table I, which highlights the superiority of the feature fusion approach over individual techniques.

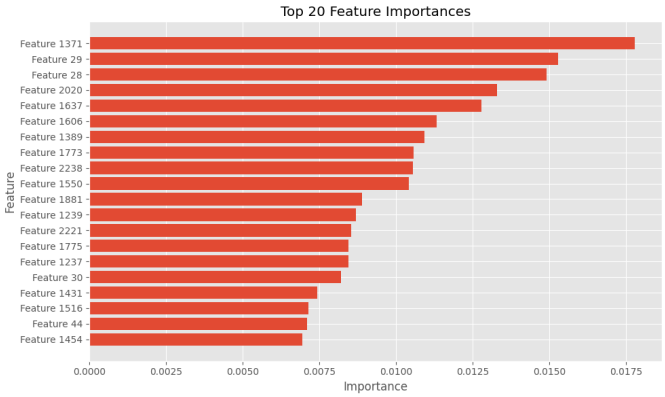

Fig. 6. Feature Importance Analysis from Random Forest Classifier.

TABLE I
PERFORMANCE COMPARISON OF FEATURE EXTRACTION METHODS

Method	Accuracy (%)	Precision (%)
Handcrafted Features	85.4	83.2
Deep Features	91.8	90.1
Feature Fusion	**95.2**	**94.7**

The accuracy improvements achieved by the feature fusion approach can be attributed to its ability to integrate domain-specific and high-level features, providing a comprehensive representation of the data.

B. ROC-AUC Analysis

To further validate the robustness of the proposed methods, Receiver Operating Characteristic (ROC) curves were analyzed. Figure 7 illustrates the comparative ROC-AUC curves for the three approaches, demonstrating the highest area under the curve (AUC) for the feature fusion approach.

Fig. 7. ROC-AUC curves for different feature extraction methods.

The ROC-AUC analysis reaffirms the feature fusion model's ability to achieve low false positive and false negative rates, which is critical in malware detection.

C. Confusion Matrix

Figure 8 presents the confusion matrix for the feature fusion approach. The minimal false positives and negatives underline the model's high reliability and effectiveness in classification tasks.

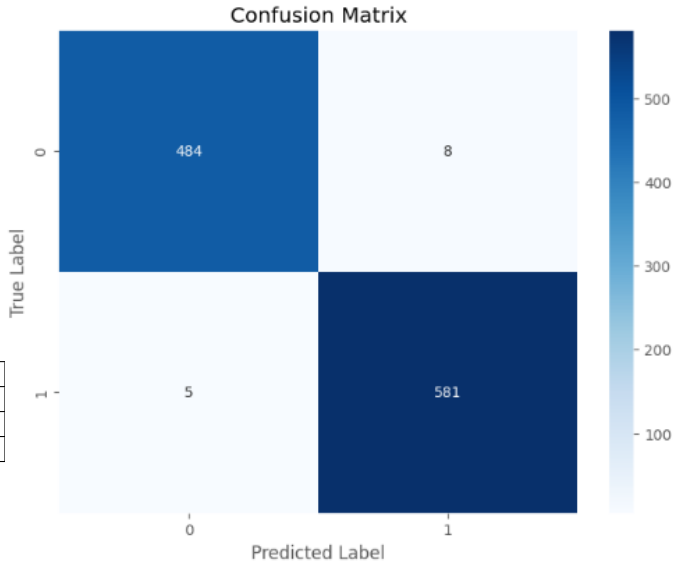

Fig. 8. Confusion matrix for the feature fusion approach.

The confusion matrix provides detailed insights into the model's performance by highlighting the true positive and true negative rates, which are consistent with the high accuracy metrics observed.

D. Comparative Study with State-of-the-Art

A comparison with existing state-of-the-art methods further validates the efficacy of the proposed framework. Table II shows the performance metrics of the proposed approach relative to previous works.

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS

Method	Accuracy (%)	Source
Nataraj et al. (2011)	92.1	[3]
Raff et al. (2018)	93.4	[1]
Proposed Approach	**95.2**	This Paper

The results indicate that the proposed feature fusion framework not only outperforms handcrafted and deep learning-based methods but also establishes a new benchmark in malware detection.

VI. CONCLUSION AND FUTURE WORK

A. Conclusion

This research highlights the potential of combining handcrafted and deep learning-based features for automated malware detection. By transforming malware binaries into grayscale images and leveraging both domain-specific insights

and transfer learning, the proposed feature fusion framework achieves state-of-the-art performance.

The integration of handcrafted features with MobileNetV2-derived features has shown to improve classification accuracy significantly. These findings underscore the value of a hybrid approach in addressing complex classification challenges.

B. Future Work

Future research directions include:

- Extending the methodology to classify a broader range of malware families to evaluate its generalization capabilities.
- Integrating explainability techniques to enhance the interpretability of the CNN models, making the decision-making process more transparent.
- Exploring the detection of advanced threats, such as fileless and obfuscated malware, to further strengthen the framework's utility in real-world cybersecurity scenarios.
- Investigating the scalability of the framework for real-time malware detection in large-scale systems.

By addressing these areas, the proposed framework can evolve into a more versatile and robust tool for combating the ever-growing threat of cyber attacks.

ACKNOWLEDGMENTS

The authors extend their gratitude to the GIK Institute for their support and resources in conducting this research.

REFERENCES

[1] E. Raff et al., "Malware detection by eating a whole EXE," *arXiv preprint arXiv:1809.04548*, 2018.

[2] J. Saxe and K. Berlin, "Deep neural network-based malware detection using two-dimensional binary program features," *22nd USENIX Security Symposium*, 2015.

[3] L. Nataraj et al., "Malware images: Visualization and automatic classification," *8th Intl. Symposium on Visualization for Cyber Security*, 2011.

[4] I. Goodfellow et al., *Deep Learning*, MIT Press, 2016.

[5] S. Zhao, X. Sun, and J. Xu, "A survey of deep learning-based network anomaly detection," *IEEE Access*, vol. 7, pp. 124604–124630, 2019.

[6] F. Rustam, M. A. Siddique, H. U. R. Siddiqui, S. Ullah, A. Mehmood, I. Ashraf, and G. S. Choi, "Wireless capsule endoscopy bleeding images classification using CNN based model," IEEE Access, vol. 9, pp. 33675–33688, 2021.

[7] J. Singh, D. Thakur, F. Ali, T. Gera, and K. S. Kwak, "Deep feature extraction and classification of Android malware images," Sensors, vol. 20, no. 24, p. 7013, Dec. 2020.

[8] P. Zegzhda, D. Zegzhda, E. Pavlenko, and G. Ignatev, "Applying deep learning techniques for Android malware detection," in Proc. 11th Int. Conf. Secur. Inf. Netw., Sep. 2018, pp. 1–8.

[9] M. Ganesh, P. Pednekar, P. Prabhuswamy, D. S. Nair, Y. Park, and H. Jeon, "CNN-based Android malware detection," in Proc. Int. Conf. Softw. Secur. Assurance (ICSSA), Jul. 2017, pp. 60–65.

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

[11] X. Xiao and S. Yang, "An image-inspired and CNN-based Android malware detection approach," in Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE), Nov. 2019, pp. 1259–1261.

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

[13] A. S. de Oliveira and R. J. Sassi, "Chimera: An Android malware detection method based on multimodal deep learning and hybrid analysis," TechRxiv, Dec. 2020, doi: 10.36227/techrxiv.13359767.v1.

[14] D. Thakur, "Classification of Android malware using its image sections," Int. J. Adv. Trends Comput. Sci. Eng., vol. 9, no. 4, pp. 6151–6155, Aug. 2020.

[15] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image," J. Ambient Intell. Humanized Comput., vol. 11, pp. 1–10, Jun. 2020.

[16] J. Singh, T. Gera, F. Ali, D. Thakur, K. Singh, and K.-S. Kwak, "Understanding research trends in Android malware research using information modelling techniques," Comput., Mater. Continua, vol. 66, no. 3, pp. 2655–2670, 2021.

[17] T. Gera, J. Singh, D. Thakur, and P. Faruki, "A semi-automated approach for identification of trends in Android ransomware literature," in Machine Learning for Networking, É. Renault, S. Boumerdassi, and P. Mühlethaler, Eds. Cham, Switzerland: Springer, 2021, pp. 265–283.

[18] D. Thakur, T. Gera, and J. Singh, "Android anti-malware techniques and its vulnerabilities: A survey," in Smart Innovations in Communication and Computational Sciences. Singapore: Springer, 2019, pp. 315–328.

[19] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of ResNet and GoogleNet models for malware detection," J. Comput. Virol. Hacking Techn., vol. 15, no. 1, pp. 29–37, Mar. 2019.

[20] L. Shiqi, T. Shengwei, Y. Long, Y. Jiong, and S. Hua, "Android malicious code classification using deep belief network," KSII Trans. Internet Inf. Syst., vol. 12, no. 1, pp. 454–475, 2018.