Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Spatio-Temporal Video Framework for Android Malware Detection using 3D Convolutional Neural Networks

ARINDAM ROY

arindamroy.aiem@jisgroup.org

Abacus Institute of Engineering & Management, Mogra, Hooghly, West Bengal

BISHAL DAS

Abacus Institute of Engineering & Management, Mogra, Hooghly, West Bengal

Research Article

Keywords: Android Malware Detection, 3D-CNN, Static Analysis

Posted Date: November 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11451263/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Spatio-Temporal Video Framework for Android Malware Detection using 3D Convolutional Neural Networks

***Abstract*—Recent research on Android malware detection has pivoted toward Deep Neural Networks (DNNs) due to their superior accuracy and automated feature-extraction capabilities. However, current image-based DNNs typically rely on resizing raw binaries to fixed dimensions, a normalization process that causes significant information loss and obscures fine-grained malicious patterns, degrading model generalization. This paper proposes a novel spatio-temporal framework that utilizes a Byte-to-Video encoding scheme to preserve the full structural integrity of the analyzed application binary file. Our scheme partitions the binary data into a sequence of high-resolution image frames, which are then processed by a 3D Convolutional Neural Network (3D-CNN) to capture both spatial features and inter-frame dependencies. By treating the application binary as a video, our model effectively identifies malicious patterns that span across the sequential logic of the bytecode without the degradation inherent in 2D normalization. Experimental evaluation on our newly curated dataset, comprising over 22 thousand samples, demonstrates that our approach achieves a 0.02 improvement in F1-score over traditional schemes. Furthermore, validation on the CICMalDroid dataset yields an average true positive rate of 0.97 across diverse malware families, representing a 0.04 improvement over existing approaches in the literature.**

***Index Terms*—Android Malware Detection, 3D-CNN, Static Analysis.**

I. Introduction

As of late 2025, Android remains the most widely used mobile operating system, accounting for approximately 72% of the global market share [1]. This extensive user base, encompassing nearly 3.9 billion active devices, makes the platform a primary target for cybercriminals, as the vast majority of mobile malware is developed specifically for the Android ecosystem. In 2025 alone, security researchers detected over 815 thousand new unique malicious installation packages, contributing to a total of more than 14 million blocked attacks throughout the year [2]. Given this persistent and evolving threat landscape, recent research has focused on developing new Android malware detection techniques, particularly those leveraging Machine Learning (ML) and behavioral analysis to safeguard Android devices [3].

Current Android malware detection strategies have increasingly leveraged Deep Neural Networks (DNNs) in both static and dynamic analysis paradigms [4]. Static analysis involves inspecting the application's bytecode, manifest files, and structural properties without executing the code, whereas dynamic analysis characterizes malware by monitoring behavioral patterns, such as system calls and network traffic, during runtime in a sandbox environment. Due to its lower computational overhead and lack of complex execution requirements, static analysis remains the more prevalent approach in the current literature. In general, static-based schemes represent malware as image data to leverage the feature-extraction capabilities of computer vision architectures [5]. In this workflow, the Dalvik Executable (`classes.dex`) file, which contains the application's compiled code, is converted into a binary stream and reshaped into a two-dimensional matrix, effectively transforming the raw bytecode into a grayscale or RGB image. This image-based representation enables DNNs to capture spatial correlations and structural signatures indicative of malicious intent, paving the way for automated threat detection without manual feature engineering.

A significant challenge in image-based malware detection arises from the inherent variability in the size of `classes.dex` files, which results in generated images of varying dimensions [6]. To accommodate the fixed-size input requirements of standard DNN architectures, traditional approaches typically employ resizing techniques to standardize these images to a uniform resolution [7]. However, this normalization process often results in the loss of structural information and fine-grained byte patterns, particularly in very large or highly complex binaries. Such information loss can severely degrade the model's ability to capture discriminative features, ultimately undermining its generalization capabilities and reducing detection accuracy on previously unseen malware variants. One potential approach to address these limitations is the use of 3D Convolutional Neural Networks (3D-CNNs), which enable the integration of multiple images to capture structural dependencies [8]. Despite their potential to preserve spatial information typically lost during 2D resizing, such 3D-based architectures remain largely unexplored in the current Android malware detection literature.

Notwithstanding, existing Android malware datasets frequently provide pre-processed Android Application Package (APK) data solely in fixed-resolution image formats [6]. By providing only resized representations, these benchmarks overlook the structural characteristics of the original binaries and inadvertently strip away the dimensional variance essential

for analysis. This data-level constraint significantly hinders the development and evaluation of variable-sized detection strategies, as researchers are often prevented from accessing the raw spatial distributions necessary to implement architectures that could otherwise adapt to the heterogeneous nature of `classes.dex` files.

Contribution. In light of this, this paper proposes a new spatial-aware Android malware detection model, implemented in three ways. First, the analyzed APK is converted into a sequence of multiple images derived from the associated `classes.dex` file, preserving the original byte distributions. Second, these generated images are structured into a video format to encapsulate the comprehensive spatial behavior of the `classes.dex`'s code. Third, the resulting video is processed by a 3D-CNN that classifies malware by considering the complete behavioral profile in a spatio-temporal context, thereby enhancing the model's generalization capabilities across diverse and complex samples. Our main insight lies in treating the Android malware binaries as temporal sequences by encoding them into a video-based representation for 3D-CNN analysis, effectively leveraging the additional dimensionality to bypass the information bottlenecks inherent in traditional 2D resizing.

In summary, our main contributions are:

- A new Android malware dataset represented in a video format extracted from the `classes.dex` files. The resulting dataset was collected considering a 4-year interval, with 22, 058 APK samples.
- A new spatial-aware Android malware detection model implemented through a 3D-CNN scheme. Our proposed model improves F1-Score by up to 0.03 in our dataset, and increases the average true-positive of malware family detection by 0.04 in a benchmark dataset compared to the literature.

Roadmap. The remainder of this paper is organized as follows. Section II overviews the traditional image-based Android malware detection strategy. Section III overviews the related works. Section IV introduces our proposed scheme, while Section V evaluates its performance. Finally, Section VI concludes our work.

II. Preliminaries

A. Image-based Android Malware Detection

The traditional implementation of image-based Android malware detection begins with extracting the component to be analyzed. Here, a typical approach lies in analyzing the `classes.dex` file, which contains the compiled Android Dalvik bytecode [6]. This binary data is converted into a one-dimensional vector of byte values and subsequently reshaped into a two-dimensional matrix, effectively representing the application's code as a grayscale or RGB image. Because DNN architectures generally require a fixed-size input (e.g., 224×224 pixels), the generated images are resized. This normalization step ensures compatibility with standard convolutional layers, allowing the model to automatically extract spatial features and structural signatures indicative of malicious behavior from the visual representation of the bytecode. The typical DNN training process requires partitioning the dataset into three distinct subsets: training, validation, and testing [4]. The training set is utilized to optimize the network's weights and biases by minimizing a predefined loss function through backpropagation, enabling the model to learn discriminative patterns from the malware samples. Concurrently, the validation set serves as an independent benchmark during training, used to tune hyperparameters and monitor for overfitting, ensuring the model generalizes well to unseen data. Finally, the test set is reserved for evaluating the fully trained model, providing a definitive assessment of its classification performance and predictive accuracy on completely novel Android applications.

B. The Generalization Challenge

A fundamental challenge to generalizing current image-based Android malware detection schemes lies in the rigid input requirements of standard DNN architectures [9]. Because these models require a fixed input resolution, researchers must resize the variable-dimensional images extracted from `classes.dex` files to a uniform scale. This normalization process frequently results in the loss of fine-grained structural artifacts or the introduction of noise through image resizing, particularly for very large binaries or those belonging to complex, sophisticated malware families [8]. Given the vast and heterogeneous landscape of Android threats, such information loss hampers the model's ability to learn robust, invariant features across diverse families, leading to a significant degradation in generalization performance when the system encounters novel variants with distinct structural compositions that were obscured during resizing.

III. Related Works

Over the past years, several works have proposed highly accurate Android malware detection schemes [4]. In general, recent strategies rely on DNNs due to their typically higher reported accuracies. S. Dong *et al.* [10] propose a D-CNN framework that integrates normalized permission features with API call graphs into a multiscale representation, outperforming individual deep learning models. However, the reliance on rigid feature normalization and dimensionality reduction, which use a fixed input format, may fail to generalize across the structural diversity of evolving malware families. H. Li *et al.* [11] employ uncertainty estimation to train a correction model that identifies and rectifies potential misclassifications in the underlying DNN, significantly improving accuracy on both emerging and adversarial samples. Similarly, despite its focus on uncertainty, the approach remains constrained by a reliance on fixed-format historical samples and a training process that may still fail to generalize. I. Haq *et al.* [12] proposed a hybrid mechanism that integrates DNNs and BiLSTM to capture both spatial and sequential features, demonstrating superior detection accuracy and time efficiency

compared to benchmark algorithms. However, the architecture's reliance on standard datasets and traditional hybrid layers introduces a vulnerability to fixed-input constraints. Haodong Li *et al.* [13] integrates uncertainty estimation with unsupervised DNN anomaly detection to identify mislabeled samples, significantly outperforming existing noise reduction methods. However, the framework focuses primarily on label refinement within established datasets, potentially inheriting the lack of generalization inherent to the fixed-input DNN architectures it seeks to cleanse. X. Zhang *et al.* [14] propose a two-stage framework that utilizes DNNs and PCA to extract and fuse features from encrypted traffic, employing a cascaded deep forest to adaptively classify malware across varying sample scales. However, reliance on traffic payload splicing and traditional feature enhancement techniques suggests a dependence on specific packet structures, which may limit generalization.

To address the variable size of analyzed Android malware, a typical approach relies on Recurrent Neural Network (RNN) schemes. M. U. Tanveer *et al.* [15] integrate LSTM and GRU networks to capture sequential dependencies in behavioral data, enabling the detection of sophisticated malware that evades traditional signature-based methods. Their approach can address the spatial characteristics; however, it relies on dynamic-based detection and manually crafted features. P. Musikawan *et al.* [16] utilize an ensemble-based DNN where predictive outputs from all hidden layers are blended, and the first layer employs multiple RNNs for feature learning, resulting in superior performance across standard classification metrics. The model addresses the variable size of analyzed APKs, however the authors evaluate their scheme in a single dataset with a limited number of malware families. H. D. Misalkar *et al.* [17] integrate multiple LSTM networks to analyze temporal API call patterns. Their proposed scheme improves accuracy on the benchmark dataset; however, the authors rely on a dynamic-based monitoring approach, which may hinder implementation.

As a result, recent literature has extensively explored various DNN architectures and hybrid feature extraction techniques to improve detection accuracy. However, a persistent challenge remains the reliance on fixed-input formats and pre-processed datasets, which leads to a significant loss of structural information and hinders the model's generalization capabilities when faced with the variable sizes of modern Android malware.

IV. Proposed Model

To address the aforementioned challenge, we propose a new video-based Android malware detection model. Our proposed scheme is shown in Figure 1 and implemented in three ways. First, the *Image Building* module extracts the `classes.dex` file from the analyzed APK and converts it into a variable-sized image format via a direct byte-to-pixel mapping. Second, the *Video Building* module structures these generated images into a sequential video representation, effectively capturing the complete spatial distribution of the application's code without the information loss typical of resizing methods. Finally, the *Classification* module uses a 3D-CNN architecture to process video data, enabling the model to learn spatiotemporal features and accurately classify the APK as *malware* or *goodware*. Our main insight is to treat Android malware binaries as temporal sequences by encoding them into a video-based representation for 3D-CNN analysis. By leveraging the additional dimensionality of 3D convolutions, the proposed model effectively bypasses the information bottlenecks inherent in traditional 2D resizing, preserving fine-grained structural artifacts that are typically discarded and significantly improving the model's generalization capabilities across diverse malware families.

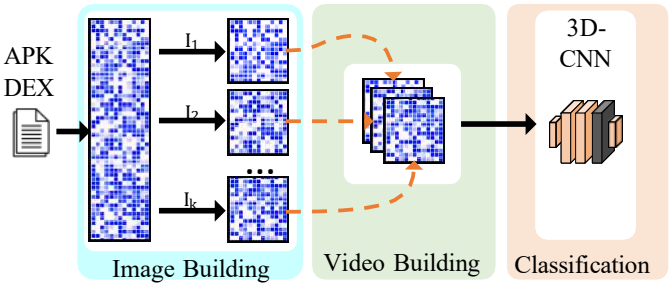


Fig. 1: Overview of proposed Android malware detection scheme. Each analyzed APK is represented through multiple DEX images, and its associated images are represented as a video. The resulting video is then classified by a 3D-CNN.

The next subsections further describe our proposed scheme, including the modules that implement it.

A. Image Building

Conducting image-based classification of Android malware by analyzing the structural patterns of the `classes.dex` file typically necessitates resizing the generated images to meet the fixed-input requirements of standard DNN architectures. This normalization process inherently causes significant information loss, as byte-level details are discarded or distorted, ultimately undermining the model's generalization capabilities across heterogeneous malware families. To address this challenge, our approach extracts multiple images from the binary data, preserving the original spatial distribution and dimensional integrity of the code (*Image Building*, Fig. 1).

Let $\mathbf{D}$ be an analyzed `classes.dex` file, represented as a finite sequence of bytes $\mathbf{D} = \{b_1, b_2, \ldots, b_n\}$, where n denotes the file size. Our objective is to transform this binary sequence into a set of images $\mathbf{I} = \{I_1, I_2, \ldots, I_k\}$, where each image I_j captures a specific segment of the bytecode without the information loss associated with global resizing. To achieve this, $\mathbf{D}$ is partitioned into k contiguous blocks, each mapped to a fixed-dimensional matrix $I_j \in \mathrm{R}^{H \times W}$ via a direct byte-to-pixel conversion, where H and W denote the image height and width, respectively. By generating this set of images, we preserve the local structural dependencies and spatial characteristics of the original file, providing a high-fidelity input for subsequent spatio-temporal analysis.

As an example, let us consider an analyzed `classes.dex` file D with a total size of ≈ 2 MB, which corresponds to $2,097,152$ bytes. Given a target resolution of 224×224 pixels for each image I_j, every image in the set captures $50,176$ bytes of the original binary data. In this instance, the file is partitioned into $k \approx 42$ segments, yielding a sequence $\mathbf{I}$ of 41 full-resolution images followed by one final image containing the remaining bytes. This structured decomposition ensures that the entire 2 MB binary is preserved and mapped into a sequential format, allowing the classification model to process the complete structural footprint of the malware without the resolution-based degradation typically found in single-image implementations.

B. Video Building

To conduct classification for each analyzed APK, the set of generated images $\mathbf{I}$ must be aggregated into a single representation that enables the model to evaluate the entire application. To achieve this task, we structure the generated images as a sequential video (*Video Building*, Fig. 1). Our transformation converts the spatial data into a spatio-temporal volume, where the sequence of images represents the progression of the `classes.dex` file. By treating the APK as a video, we can apply 3D-CNN architectures to capture both the intra-image spatial features and the inter-image temporal dependencies, resulting in a single classification output that characterizes the entire binary. Formally, the image set $\mathbf{I} = \{I_1, I_2, \ldots, I_k\}$ is arranged into a 3D tensor $\mathbf{V} \in \mathbb{R}^{k \times H \times W}$, where k represents the number of frames (depth), and $H \times W$ the spatial dimensions of each frame.

C. Classification

Our classification process is designed to analyze the entire structural footprint of an Android application by processing the spatial dependencies inherent in the sequenced bytecode. As a result, we treat the set of images $\mathbf{I}$ as a continuous video volume $\mathbf{V}$, and we employ a 3D-CNN to perform a single, holistic classification (*Classification*, Fig. 1). This enables the model to extract hierarchical features that represent the complex, multi-frame patterns characteristic of sophisticated malware families.

Formally, for a 3D convolutional layer, the value of a unit at position (x, y, z) in the j-th feature map of the i-th layer is denoted as $v_{i,j}^{x,y,z}$ and is computed as:

$$v_{i,j}^{x,y,z} = \sigma\left(\sum_{m} \sum_{p,q,r} w_{i,j,m}^{p,q,r} \cdot v_{(i-1),m}^{(x+p),(y+q),(z+r)} + b_{i,j} \right) \quad (1)$$

where m indexes the feature maps in the $(i-1)$-th layer, $w_{i,j,m}^{p,q,r}$ is the (p, q, r)-th value of the kernel connected to the m-th feature map, and $b_{i,j}$ is the bias. Through successive 3D convolutions and pooling, the network reduces the volume into a high-level feature vector, which is passed through fully connected layers to produce the final classification $y \in \{0, 1\}$. This architecture ensures that the detection of malicious intent is predicated on the global structural organization of the `classes.dex` file rather than isolated code fragments.

The 3D-CNN training procedure employs a supervised learning approach, optimizing the architecture to minimize the variance between predicted probabilities and ground truth labels across a diverse dataset of benign and malicious APKs. Parameters are updated iteratively via backpropagation using the Binary Cross-Entropy (BCE) loss function, which is defined for a batch of N samples as:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2)$$

where y_i is the ground truth and $\hat{y}_i$ is the predicted probability. By minimizing this loss, the model effectively learns to distinguish the complex spatial signatures of malware within the video-encoded bytecode.

D. Discussion

Our proposed model aims to address a limitation in current image-based Android malware detection caused by the forced normalization of varying binary sizes into fixed 2D dimensions. To achieve such a goal, we transition from a static image representation to a spatial video volume. Our approach preserves the full structural integrity of the `classes.dex` file, ensuring that no byte-level information is sacrificed to meet architectural constraints. The integration of 3D-CNNs allows the system to capture the sequential continuity of code execution patterns as they manifest across the partitioned image frames. This multidimensional perspective provides a more comprehensive understanding of the malware's organizational logic, resulting in a detection mechanism that is inherently more robust against multiple malware families than traditional 2D deep learning frameworks.

V. Evaluation

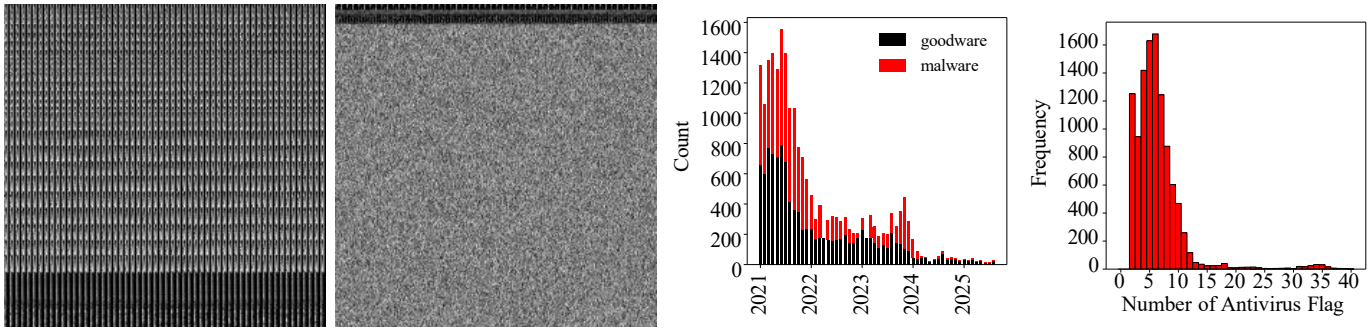
Our conducted set of experiments aims at answering the following Research Questions (RQs):

- **RQ1.** *What is the Android malware detection performance of traditional 2D-CNN strategies?*
- **RQ2.** *How does our proposed scheme improve Android malware detection accuracy?*
- **RQ3.** *What is the accuracy improvement of our scheme versus traditional approaches?*

The next subsections further evaluate our scheme and describe our built dataset.

A. A Realistic Android Malware Dataset

A significant limitation in the current literature is that existing Android malware datasets do not provide APKs in a variable-sized image format, which hinders the direct evaluation of schemes designed to handle structural diversity without information loss. To address this challenge, we constructed a new dataset that captures the complete behavior of the applications by processing their associated `classes.dex`



(a) 1st Goodware Sample Frame (b) 1st Malware Sample Frame (c) APK Sample distribution in time (d) Distribution of antivirus flagged

Fig. 2: Sample images built by our proposed prototype according to a 224*x*224 image format, and the distribution of samples in our dataset.

files. This was achieved through a direct byte-to-pixel conversion of the raw binary data, ensuring that each analyzed APK `classes.dex` file is represented by a set of generated images that maintain a consistent 224×224 resolution, but can be represented by multiple images (see Fig. 1). By adopting this approach, we generate a variable number of images for each analyzed APK, depending on its original file size, preserving the full structural integrity of the bytecode and providing a high-fidelity representation suitable for spatio-temporal classification. Figures 2a and 2b show the first image samples generated from a *goodware* and *malware* APKs in our dataset. As a result, our dataset provides a high-fidelity structural mapping of the `classes.dex` file, eliminating the information loss inherent in traditional binary-to-image normalization.

Our built dataset comprises a total of 22, 058 APK samples, consisting of 11, 107 *goodwares* and 10, 951 *malware* applications. These samples were curated from the AndroZoo repository over a five-year interval to ensure a longitudinal representation of evolving malware characteristics and legitimate software updates. Figure 2c shows the time distribution of the APK samples collected in our dataset. For ground-truth verification, we conducted a labeling process by submitting each sample to VirusTotal. An application was labeled as malware if it was flagged by at least two distinct antivirus engines, thereby mitigating the risk of false positives from individual scanners and ensuring the reliability of the dataset for training and validation. Figure 2d shows the distribution of the number of antivirus flags for each *malware* sample in our dataset.

We selected two widely utilized architectures, ResNet-18 and DenseNet121, implemented in both 2D and 3D configurations, to evaluate the efficacy of our spatial approach. The models were developed using PyTorch v2.1.0, with 3D variants adapted to process the sequential video volumes generated from the `classes.dex` binaries. Training was conducted over 100 epochs using the Adam optimizer with a primary learning rate of 1×10^{-5} and a batch size of 32. To ensure convergence and prevent overfitting, we employed a weight decay of 1×10^{-5} and a learning rate scheduler that reduced the rate by a factor of 0.1 upon detecting a plateau. Standard architectural parameters were maintained for consistency, including bottleneck layers and global average pooling before the final SoftMax classification layer.

TABLE I: Performance Comparison of 2D and 3D Architectures in our built dataset

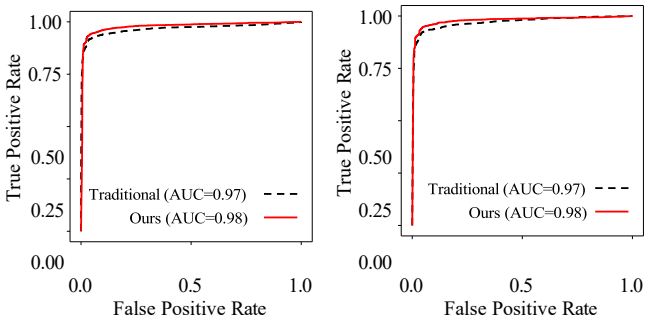
Approach	TPR	TNR	F1	AUC
Traditional 2D-CNN (DenseNet)	0.95	0.93	0.93	0.96
Ours 3D-CNN (DenseNet)	0.95	0.96	0.95	0.98
Traditional 2D-CNN (ResNet)	0.95	0.93	0.93	0.96
Ours 3D-CNN (ResNet)	0.95	0.95	0.95	0.98

Our resulting dataset was partitioned into training, validation, and test sets, comprising 60%, 20%, and 20% of the total samples, respectively. This split was performed using a stratified approach to ensure a balanced distribution of *goodware* and *malware* samples across all subsets. The selected schemes were evaluated on the test dataset using their measured True Positive Rate (TPR), True Negative Rate (TNR), and F1-Score. The TPR represents the proportion of malware samples correctly identified, while the TNR denotes the ratio of benign samples accurately classified. Finally, the F1-Score was computed as the harmonic mean of Precision and Recall.

B. Image-based Android Malware Detection

Our first experiment aims at answering *RQ1*, and investigates the accuracy performance of traditional 2D-CNN strategies. To achieve this goal, we assess the classification performance of 2D-CNNs implemented using the ResNet and DenseNet architectures on our newly constructed dataset. Here, we use the traditional image resizing approach described in the literature, where the resulting `classes.dex` file image is resized to 224 × 224. The goal is to establish a performance baseline for traditional image-based detection methods that rely on single-image representations.

Table I shows the classification performance of traditional 2D-CNN strategies. As observed, both the ResNet and DenseNet architectures achieved an F1-score of 0.93. While these results represent a high baseline for classification, the



(a) ResNet (b) DenseNet

Fig. 3: ROC curve of our scheme *vs.* traditional 2D-CNN in our dataset.

performance ceiling is attributable to the inherent limitations of the resizing approach used in 2D-CNN pipelines. To accommodate the fixed-input requirements of these architectures, the original `classes.dex` representation is subjected to interpolation, which inevitably obscures fine-grained byte-level patterns.

Our second experiment aims at answering *RQ2*, and investigates the accuracy performance of our proposed model in our dataset. To achieve this goal, we apply the same architectures used previously, but in a 3D-CNN configuration, utilizing our proposed scheme as shown in Figure 1.

Table I shows the classification performance of our proposed approach. The results indicate consistent improvement across all metrics, with a 0.02 increase in F1-Score for both the ResNet and DenseNet configurations compared to their 2D counterparts. This performance gain is primarily attributed to eliminating the information bottleneck inherent in single-image resizing. By representing the `classes.dex` file as a sequence of high-resolution images, the 3D-CNN can process the raw bytecode without the structural degradation caused by interpolation. Furthermore, the 3D convolutional kernels successfully capture the spatio-temporal continuity of the file, allowing the model to recognize malicious patterns that span multiple frames—features that are typically lost or conflated when a large binary is compressed into a static 2D plane. Figure 3 shows the Receiver Operating Characteristic (ROC) curve of our proposed scheme *vs.* the traditional 2D-CNN approach. Similarly, our proposed scheme improves the Area Under the Curve (AUC) in both configurations when compared to the traditional approach.

Finally, to answer *RQ3* we assess the performance of our scheme in a widely used benchmark dataset in the literature. To achieve such a goal, we evaluate our scheme in the CICMalDroid dataset [18], which comprises 17, 341 samples categorized into four distinct malware families: Adware, Banking, Riskware, and SMS. To overcome the limitations associated with image resizing and ensure compatibility with our spatio-temporal framework, we downloaded the raw malware samples and executed our processing pipeline to generate a video vol-ume from each `classes.dex` file associated with the APKs in the dataset (*Video Building*, Fig. 1). This procedure enables a direct comparison between our high-fidelity sequential en

Adware	0.98	0.98	0.93
Banking	0.96	0.97	0.94

TABLE II: Performance Comparison of 2D and 3D Architectures in the CICMalDroid dataset

Approach	TPR	TNR	F1	AUC
Ours 3D-CNN (DenseNet)	0.97	0.97	0.97	0.99
Ours 3D-CNN (ResNet)	0.96	0.97	0.97	0.99
DNN [19]	0.93	0.94	0.93	-

TABLE III: Malware Family Detection Performance Comparison in CICMalDroid dataset

Malware Family	Ours		DNN [19]
	DenseNet	ResNet	
Riskware	0.94	0.93	0.93
SMS	1.00	1.00	0.94
Avg.	0.97	0.97	0.93

coding and traditional static representations in a standardized benchmarking environment. Further, we compared our scheme with the DNN approach proposed by M. Rashid *et al.* [19], which also uses a 2D-CNN with an image-resizing approach. Table II shows the accuracy performance of our scheme on CICMalDroid dataset. Consistent with our previous experiments, the proposed scheme demonstrates superior classification performance compared to traditional approaches reported in the literature. Notably, our model achieves a 0.04 improvement in F1-Score over the method proposed by M. Rashid *et al.* [19]. We also investigate the accuracy improvement on a per-family basis. Table III shows the accuracy performance of our scheme for each malware family available in CICMalDroid dataset. Our proposed model achieves an average accuracy of 0.97 across the families, whereas the traditional approach reaches only 0.93. This consistent performance gain across distinct malicious categories, including Adware, Banking, Riskware, and SMS, demonstrates our model's superior generalization capabilities. By leveraging the spatial features of the bytecode, the scheme maintains high detection precision regardless of the specific behavioral traits or code structures unique to different malware families, effectively addressing the limitations inherent in static, resolution-dependent classification methods.

VI. Conclusion

In this paper, we presented a novel spatial framework for Android malware detection that addresses the challenge of information loss in image-based binary analysis. By transitioning from static 2D resizing to a Byte-to-Video encoding scheme, our approach preserves the full structural integrity of the `classes.dex` file, allowing a 3D-CNN architecture to capture complex dependencies across sequential bytecode frames. Experimental results on both our new large-scale curated dataset and the benchmark CICMalDroid dataset demonstrate that our model consistently outperforms traditional 2D-DNN strategies, achieving significant improvements in F1-score and demonstrating superior generalization across diverse malware families.

Future research will focus on extending this spatio-temporal framework to integrate attention mechanisms to identify the specific code segments most critical for classifying complex malware families.

REFERENCES

[1] Willie, "Android global market share statistics 2026," 2026. [Online]. Available: https://commandlinux.com/android/android-global-market-share-statistics/

[2] S. List, "The mobile threat landscape in 2025," 2026. [Online]. Available: https://securelist.com/mobile-threat-report-2025/119076//

[3] H. Haidros Rahima Manzil and S. Manohar Naik, "Detection approaches for android malware: Taxonomy and review analysis," *Expert Systems with Applications*, vol. 238, p. 122255, Mar. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2023.122255

[4] L. Shu, S. Dong, H. Su, and J. Huang, "Android malware detection methods based on convolutional neural network: A survey," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 5, p. 1330–1350, Oct. 2023. [Online]. Available: http://dx.doi.org/10.1109/TETCI.2023.3281833

[5] J. Geremias, E. K. Viegas, A. O. Santin, A. Britto, and P. Horchulhack, "Towards multi-view android malware detection through image-based deep learning," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, May 2022, p. 572–577. [Online]. Available: http://dx.doi.org/10.1109/IWCMC55113.2022.9824985

[6] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, p. 107138, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2020.107138

[7] S. Dong, L. Shu, and J. Huang, "A novel detection method for unknown android malware via image representation," *Journal of Information Security and Applications*, vol. 99, p. 104451, 2026. [Online]. Available: http://dx.doi.org/10.1016/j.jisa.2026.104451

[8] M. M. Yapici, "3dmaldroid: A novel 3d image based approach for android malware detection and classification," *Computers and Electrical Engineering*, vol. 127, p. 110542, Oct. 2025. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2025.110542

[9] P. N. Duy and N. T. Cam, "uitobfamc: Obfuscated android malware classification using deep learning on multi-feature information approach," *Information Sciences*, vol. 720, p. 122528, Dec. 2025. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2025.122528

[10] S. Dong, L. Shu, and S. Nie, "Android malware detection method based on cnn and dnn bybrid mechanism," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 5, p. 7744–7753, May 2024. [Online]. Available: http://dx.doi.org/10.1109/TII.2024.3363016

[11] H. Li, X. Cheng, L. Wang, and H. Wang, "Towards improved dnn-based android malware detection via uncertainty estimation," *ACM Transactions on Software Engineering and Methodology*, Apr. 2026. [Online]. Available: http://dx.doi.org/10.1145/3807456

[12] I. U. Haq, T. A. Khan, and A. Akhunzada, "A dynamic robust dl-based model for android malware detection," *IEEE Access*, vol. 9, p. 74510–74521, 2021. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2021.3079370

[13] H. Li, X. Cheng, G. Zhang, G. Xu, G. Xu, and H. Wang, "Mitigating emergent malware label noise in dnn-based android malware detection," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, p. 1136–1159, 2025. [Online]. Available: http://dx.doi.org/10.1145/3715769

[14] X. Zhang, J. Wang, J. Xu, and C. Gu, "Detection of android malware based on deep forest and feature enhancement," *IEEE Access*, vol. 11, p. 29344–29359, 2023. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2023.3260977

[15] M. Usama Tanveer, K. Munir, A. Alabdulatif, A. R. Najdawi, and R. H. Jhaveri, "Malware-seqguard: An approach utilizing lstm and gru for effective detection of evolving malware in android environments," *IEEE Access*, vol. 13, p. 117355–117373, 2025. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2025.3585605

[16] P. Musikawan, Y. Kongsorot, I. You, and C. So-In, "An enhanced deep learning neural network for the detection and identification of android malware," *IEEE Internet of Things Journal*, vol. 10, no. 10, p. 8560–8577, May 2023. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2022.3194881

[17] H. D. Misalkar and P. Harshavardhanan, "Tdbamla: Temporal and dynamic behavior analysis in android malware using lstm and attention mechanisms," *Computer Standards & Interfaces*, vol. 92, p. 103920, Mar. 2025. [Online]. Available: http://dx.doi.org/10.1016/j.csi.2024.103920

[18] S. Mahdavifar, A. F. Abdul Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning," in *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)*. IEEE, Aug. 2020, p. 515–522. [Online]. Available: http://dx.doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094

[19] M. U. Rashid, S. Qureshi, A. Abid, S. S. Alqahtany, A. Alqazzaz, M. ul Hassan, M. S. Al Reshan, and A. Shaikh, "Hybrid android malware detection and classification using deep neural networks," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, Mar. 2025. [Online]. Available: http://dx.doi.org/10.1007/s44196-025-00783-x