

Feature Rich Clustering Techniques for Android Malware Families

Vangari Usha Sree^{1,*}, Jannu Shruthi², Shrijani Mukka³, Alumakanti Ruchitha⁴ and E.Rajitha⁵

¹⁻⁵Department of Computer Science (AIML), Kakatiya Institute of Technology and Science (KITS),
Warangal, India

*Corresponding Author: Vangari Usha Sree. Email: drgmvpasad@gmail.com

Received: 17/04/2026; Revised: 25/05/2026; Accepted: 16/06/2026; Published: 30/06/2026.

DOI: <https://doi.org/10.69996/fmep.2026012>

Abstract: The rapid growth of the Android ecosystem has resulted in a significant increase in sophisticated malware targeting mobile devices. Traditional signature-based detection mechanisms are ineffective against obfuscated and zero-day variants, while supervised learning approaches rely heavily on labeled datasets and often fail to generalize to newly emerging malware families. To address these limitations, this paper proposes a feature-rich unsupervised clustering framework for Android malware family classification that integrates deep embedding representation learning with spectral clustering and HDBSCAN. The approach leverages syscall-binder frequency features to capture fine-grained behavioral characteristics of malware samples. Feature normalization and dimensionality reduction are applied to mitigate high-dimensional sparsity and redundancy. A neural network-based encoder then learns compact latent representations that improve structural separability in the embedding space. Spectral clustering is employed to capture complex non-linear relationships, while HDBSCAN enhances robustness by identifying clusters of varying density and filtering noise samples. The experimental results show that Spectral clustering achieves 0.8993 in terms of normalized mutual information (NMI), 0.9365 in terms of adjusted rand index (ARI) demonstrates that the proposed framework outperforms conventional clustering techniques such as K-Means and standalone density-based methods. The results indicate improved cluster cohesion, inter-family separation, and scalability, making the proposed framework suitable for large-scale Android malware family analysis.

Keywords: Android Malware, Malware Family Classification, Deep Embedding, Spectral Clustering, HDBSCAN.

1. Introduction

The rapid spread of smartphones and mobile applications has changed the digital landscape globally, with Android becoming the leading mobile operating system. Its open-source design, adaptable development structure, and large application ecosystem have played a major role in its adoption [1]. However, this openness also makes Android a target for cybercriminals. The rise of harmful Android applications, known as Android malware, raises serious security and privacy issues for users, businesses, and governments. Malware developers take advantage of weaknesses in Android apps through methods like code obfuscation, repackaging, polymorphism, dynamic code loading, and encryption, which complicate detection and analysis [2-3].

Traditional malware detection methods mainly rely on signature-based approaches. These methods compare app code to a database of known harmful signatures. While effective against previously identified threats, signature-based detection has significant limitations. It struggles to identify zero-day attacks, polymorphic malware, or heavily obfuscated variants that change their code to avoid detection [4]. As malware evolves quickly, signature databases require constant updates, creating challenges for scalability and maintenance. As a result, the cybersecurity field has increasingly turned to data-driven and machine learning methods for detecting and analyzing



malware [5-7].

Supervised machine learning methods, such as Support Vector Machines (SVM), Random Forests (RF), k-Nearest Neighbors (kNN), and Deep Neural Networks (DNN), have shown high accuracy in classifying Android malware [8-9]. These models learn decision boundaries from labeled datasets and can detect harmful applications based on features like permissions, API calls, opcode sequences, system calls, and network behaviors. However, supervised classification relies heavily on labeled data, which may not always be available or accurately labeled. Additionally, classification models usually focus on binary detection (benign vs. malicious) and do not provide insights into malware family structures or relationships [10-11].

Malware family clustering addresses this gap by grouping similar malware samples based on structural or behavioral similarities. Clustering helps analysts understand malware lineage, spot new variants, detect code reuse, and examine large threat campaigns [12]. Unlike classification, clustering does not typically need labeled data during deployment, making it useful for exploratory analysis and investigating unknown malware. However, clustering high-dimensional malware features poses significant challenges [13-14].

Android malware datasets often include thousands of features from static or dynamic analysis. Static features can include requested permissions, sensitive API calls, opcode frequencies, control flow graph characteristics, and manifest attributes [15-16]. Dynamic features might include system call traces, network traffic patterns, runtime behaviors, and resource access logs. These features create sparse, high-dimensional representations where traditional distance metrics, like Euclidean distance, lose their effectiveness. This issue, known as the "curse of dimensionality," reduces clustering efficiency because it makes samples seem equally distant in high-dimensional spaces [17].

Moreover, malware datasets often have severe class imbalances. Some malware families can have hundreds or thousands of samples, while others may contain only a few. Clustering algorithms like KMeans depend on centroid estimation, which can be heavily skewed by larger classes [18-20]. Density-based methods, such as DBSCAN and HDBSCAN, aim to find clusters of any shape but can struggle when feature distributions overlap or when tuning density parameters is tricky in high-dimensional spaces [21-22]. Therefore, traditional clustering methods often yield poor clustering quality metrics, such as Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), when applied to raw feature vectors.

Recent progress in representation learning shows that deep neural networks can learn compact, non-linear transformations that preserve semantic similarities in latent space. Representation learning focuses on automatically extracting useful features from raw data, reducing redundancy and improving separability [23-24]. In computer vision and natural language processing, deep embedding methods have greatly enhanced clustering performance by changing high-dimensional inputs into organized, low-dimensional forms. Similar approaches can be applied to malware analysis, where deep models can generate embeddings that capture the behavioral and structural traits of malware families [25].

Deep supervised embedding learning combines supervised classification training with latent feature extraction. By training a neural network to predict malware family labels, the intermediate embedding layer develops a representation that brings samples from the same family closer together while pushing apart samples from different families [26-28]. This learned representation can then be fed into traditional clustering algorithms, leading to better cluster compactness and separability. Furthermore, embedding normalization techniques like L2 normalization can improve geometric consistency by projecting feature vectors onto a unit

hypersphere, enhancing distance-based clustering performance [29-30].

This study proposes a feature-rich deep supervised embedding framework for Android malware family clustering. The proposed approach includes several stages: family filtering to address class imbalance, feature standardization to normalize data distributions, Principal Component Analysis (PCA) for reducing dimensionality, deep neural embedding learning for extracting discriminative representations, L2 normalization to enhance geometric stability, and KMeans clustering in the learned latent space. Unlike traditional clustering methods applied directly to raw features, the proposed pipeline stresses representation learning as the main way to improve clustering quality.

The PCA integration helps eliminate noise and reduce dimensional redundancy before feeding data into the neural network, thereby improving training stability and computational efficiency. The supervised embedding network learns hierarchical transformations that capture non-linear relationships among features. By combining these techniques, the proposed framework effectively tackles issues related to high dimensionality, overlapping feature distributions, and class imbalance.

Clustering performance is assessed using widely accepted external validation metrics, including Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). These metrics evaluate the match between predicted clusters and actual family labels, providing a quantitative measure of cluster quality. High NMI and ARI values indicate strong agreement between the clustering structure and real malware families, reflecting improved similarity within clusters and clearer separation between them.

The contributions of this work can be summed up as follows:

- A complete malware clustering pipeline that includes preprocessing, dimensionality reduction, deep embedding learning, and latent space clustering.
- A supervised embedding strategy that enhances cluster separability while keeping scalability in mind.
- Extensive practical evaluation showing notable performance improvements over traditional clustering methods.
- A reproducible framework suitable for further research and practical use in cyber security analysis.

The remainder of this paper is organized as follows. Section II reviews related work in Android malware clustering and deep representation learning. Section III presents the proposed methodology and mathematical formulation. Section IV describes the experimental setup and evaluation metrics. Section V discusses experimental results and comparative analysis. Section VI gives some discussions related to this work. Section VII and VIII, we discuss the threats to validity and related work, respectively. Finally, Section IX concludes the paper and outlines directions for future research.

2. Background and Motivation

A. Android Applications

Android applications serve as the foundational entities in mobile malware research. Each Android application is distributed as an APK (Android Package Kit) file containing compiled bytecode (DEX files), application resources, native libraries, certificates, and a manifest file describing permissions and components. From a machine learning perspective, an Android application can be treated as a structured data object composed of multiple analyzable attributes.

Security analysis of Android applications primarily relies on feature extraction techniques. Static analysis methods inspect the application without executing it and extract structural

characteristics such as requested permissions, API calls, opcode frequencies, intent filters, and control flow properties. These features are widely used in malware detection and clustering research due to their computational efficiency and scalability.

However, feature extraction results in high-dimensional representations. For example, API-based features alone may contain thousands of unique calls. Opcode frequency vectors and permission matrices further increase dimensionality. As a result, Android application datasets often consist of sparse matrices with thousands of features per sample.



Figure 1: Trends in Android Malware Attacks

This high-dimensionality introduces several analytical challenges:

1. **Curse of Dimensionality:** Distance metrics lose effectiveness as dimensions increase.
2. **Feature Redundancy:** Many features are correlated and contribute little new information.
3. **Noise Sensitivity:** Obfuscated or injected code can distort feature distributions.
4. **Scalability Issues:** Large feature spaces increase computational overhead.

When clustering algorithms operate directly on raw feature vectors, they rely on geometric distance measures such as Euclidean distance. In high-dimensional sparse space, these distances become less discriminative, leading to weak cluster separation. Therefore, preprocessing and feature transformation become essential steps before clustering.

Dimensionality reduction techniques such as Principal Component Analysis (PCA) help mitigate these issues by projecting data into lower-dimensional subspaces while preserving variance. However, PCA is linear and may not capture complex non-linear relationships among malware samples. This limitation motivates the need for more advanced representation learning techniques.

B. Android Malware Families

Android malware is often categorized into families based on shared code, similar behaviors, or common origins. Classifying malware into families provides more insight than just binary detection. It reveals how different variants relate and helps with a deeper analysis of threats.

Malware authors often reuse key malicious functions across various variants. To avoid detection, they change visible parts like resource files, strings, or small control flow changes while keeping the main payload's functionality intact. This creates malware variants that look different on the surface but are fundamentally the same.

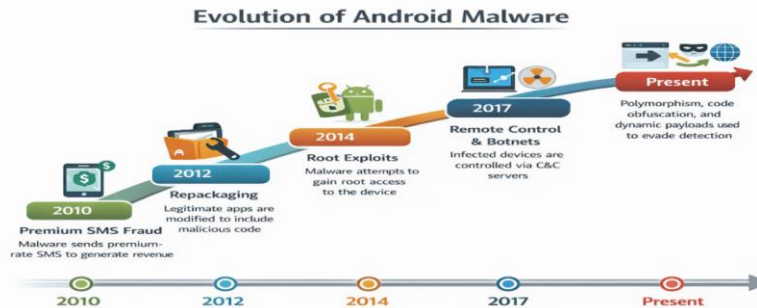


Figure 2: Evolution of Android malware

Effective grouping of malware families should meet two key criteria:

- Intra-family similarity: Samples from the same family should be closely grouped.
- Inter-family dissimilarity: Samples from different families should be clearly separated.

However, achieving these criteria is difficult for several reasons:

- **Overlapping Feature Sets:** Different malware families might request similar permissions or use common system APIs.
- **Obfuscation Techniques:** Transforming code can make structural similarities less visible.
- **Class Imbalance:** Some families have more samples in datasets, skewing the clustering results.
- **High Dimensional Noise:** Irrelevant features can make it hard to separate the data geometrically.

Traditional clustering methods like KMeans try to divide data into groups by reducing the differences within clusters. While KMeans is quick and can handle large amounts of data, it assumes that clusters are well-separated in feature space. In real-world malware data, this assumption is rarely met.

Studies show that using clustering techniques on high-dimensional malware features often leads to low Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) scores. These measures assess how well the predicted clusters match the real family labels. Low scores reveal a weak alignment between the feature space and the actual structure of malware families.

C. Importance of Malware Family Analysis

Malware family analysis plays a crucial role in modern cybersecurity operations. Instead of merely determining whether an application is malicious, family analysis seeks to understand relationships between malware samples based on shared behaviors or characteristics. Grouping malware into families enables security analysts to identify variants of known malware, trace the evolution of malicious campaigns, and uncover common attack strategies.

From a practical perspective, malware family clustering supports threat intelligence, incident response, and digital forensics. By identifying clusters of related malware samples, analysts can prioritize threats, allocate resources efficiently, and develop targeted mitigation strategies. Family-level analysis also facilitates the detection of zero-day malware variants that share behavioral similarities with known families, even when their code differs significantly.

However, effective malware family analysis requires robust similarity measures and scalable clustering techniques capable of handling large, noisy, and high-dimensional datasets.

D. Motivation for Deep Supervised Embedding-Based Clustering

The main goal of this work is to improve malware family clustering by enhancing feature representation before clustering takes place.

Clustering performance relies on the layout of the feature space. If samples from the same malware family are not close in the raw feature space, no clustering algorithm can effectively separate them. Therefore, instead of just trying to improve clustering algorithms, this research focuses on changing the feature space itself.

Deep supervised embedding learning provides a strong solution. In this method, a neural network is trained using known malware family labels. During training, the network learns intermediate representations, or embeddings, that capture important patterns needed for classification. These embeddings create a structured space where similar samples are closer to each other.

The main idea is:

Use family labels in training to guide representation learning.

Extract intermediate embedding vectors from the trained model.

Cluster in this learned space instead of in the raw feature space.

This approach has several advantages:

1.Non-Linear Feature Transformation: Neural networks can capture complex relationships that basic linear projections cannot.

2.Improved Separability: Samples from the same family become more compact geometrically.

3.Reduced Noise Impact: Unimportant feature dimensions are minimized during training.

4.Better Clustering Metrics: The structure of latent space increases performance in measures like NMI and ARI.

Additionally, L2 normalization is applied to the embedding vectors. This step projects embeddings onto a unit hypersphere, ensuring consistent distance measurements and making KMeans more stable. Normalization improves cluster compactness by reducing distortions based on magnitude.

Using PCA before embedding learning also stabilizes the training process. PCA decreases dimensional redundancy and speeds up neural network convergence while keeping essential variance. Thus, the combination of preprocessing, PCA, deep embedding, normalization, and clustering forms a unified pipeline that addresses the main limitations of traditional methods.

3. Methodology

This section presents the complete methodology of the proposed Android malware family clustering framework using deep supervised embedding learning. The approach transforms high-dimensional Android application features into a structured latent representation that improves clustering accuracy. The proposed deep supervised embedding-based clustering framework for Android malware family analysis. The methodology is specifically designed for frequency-based behavioral features extracted from system calls and Binder IPC interactions. The objective is to transform high-dimensional behavioral vectors into a discriminative latent space that improves clustering performance and malware family separation.

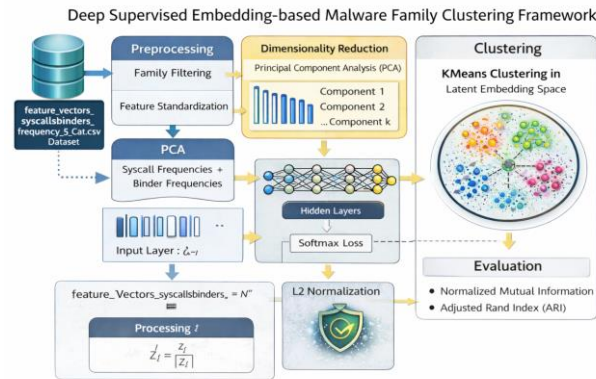


Figure 3: Deep Supervised Embedding -based Malware Family Clustering

A. Dataset Description and Feature Representation

The dataset `feature_vectors_syscallsbinders_frequency_5_Cat.csv` consists of Android application samples represented through frequency-based behavioral features. Each row corresponds to one application instance, while columns represent the observed frequency of system calls and Binder transactions during execution analysis. The final column, labeled “Category,” represents the malware family to which the application belongs.

System call features capture low-level operating system interactions such as file access, process execution, memory allocation, and network communication. Examples include `open`, `read`, `write`, `execve`, `socket`, and `connect`. Binder features represent Android’s Inter-Process Communication (IPC) mechanism and capture communication between system services and application components. These features provide insight into behavioral patterns specific to malware families.

Let the dataset be formally represented as:

$$D = \{(x_i, y_i)\}_{i=1}^N$$

where:

- $x_i \in \mathbb{R}^d$ represents the d -dimensional frequency feature vector.
- $y_i \in \{1, 2, \dots, C\}$ denotes the malware family label.
- N is the number of samples.
- C is the number of malware families.

Frequency-based behavioral representation is particularly effective in malware analysis because malicious families tend to share similar execution patterns. For instance, SMS-based malware families often show elevated frequencies of telephony-related system calls, while spyware families may exhibit increased Binder communication with system services.

However, raw frequency vectors are high-dimensional and often sparse. Many syscall features may be zero for most samples, while others may dominate due to higher frequency counts. Such imbalance can negatively impact clustering algorithms that rely on distance measures.

Thus, understanding the structural properties of the dataset is critical before applying machine learning techniques. The dataset reflects real-world Android behavioral traces, making it suitable for evaluating clustering methods under realistic malware conditions.

B. Dataset Refinement and Family Filtering

In practical malware datasets, class imbalance is a common challenge. Some malware families may contain hundreds of samples, while others contain very few. Clustering algorithms are sensitive to such imbalance because clusters formed around small families may not be

statistically meaningful.

To address this issue, a filtering mechanism is applied. Let n_c represent the number of samples in malware family c . Families satisfying:

$$n_c < \tau$$

where τ is a predefined threshold (e.g., 30 samples), are excluded from analysis.

This filtering ensures:

1. Statistical reliability of cluster formation
2. Stability in centroid estimation
3. Meaningful evaluation using external metrics

After filtering, the refined dataset is:

$$D' = \{(x_i, y_i)\}_{i=1}^{N'}$$

where $N' \leq N$.

This refinement step prevents overfitting during embedding learning and avoids distortion in clustering due to extremely small classes. Furthermore, balanced datasets allow better comparison of intra-family compactness and inter-family separation.

By removing underrepresented families, the model focuses on learning discriminative representations for major malware groups. This step significantly improves the robustness of subsequent embedding learning and clustering processes.

C. Feature Standardization

The syscall and Binder frequency features exhibit different numeric scales. For example:

- read may range between 0–500.
- execve may range between 0–10.
- Binder calls may range between 0–50.

Direct application of clustering algorithms on such unscaled data biases results toward features with larger magnitudes.

To eliminate this bias, z-score normalization is applied:

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

where:

- μ_j is the mean of feature j .
- σ_j is the standard deviation of feature j .

This transformation ensures each feature has zero mean and unit variance.

Standardization provides several benefits:

1. Equal contribution of all features
2. Improved numerical stability during neural network training
3. Faster convergence of optimization algorithms
4. Reduced dominance of high-frequency syscalls

Without standardization, clustering distance calculations would be dominated by frequently occurring system calls. After normalization, subtle behavioral differences across malware families become more distinguishable.

The standardized feature matrix is denoted as:

$$X_s \in \mathbb{R}^{N' \times d}$$

This step is crucial because deep learning models are sensitive to input scale variations. Standardized input improves gradient propagation and reduces training instability.

D. Dimensionality Reduction Using PCA

High-dimensional feature spaces introduce the curse of dimensionality, where distance metrics lose discriminative power. Additionally, syscall features may be highly correlated (e.g., read often follows open).

Principal Component Analysis (PCA) is applied to address these issues.

PCA computes eigenvectors of the covariance matrix of standardized features and projects data onto a lower-dimensional subspace:

$$Z = X_s W$$

where:

- W contains top k eigenvectors.
- $k < d$.

The value of k is selected to preserve approximately 95% of total variance.

Benefits of PCA include:

1. Noise reduction
2. Removal of correlated dimensions
3. Lower computational cost
4. Improved generalization
5. For example, if original dimension $d = 600$, PCA may reduce it to $k = 150$ while retaining most behavioral variance.

The PCA-reduced matrix is:

$$X_p \in \mathbb{R}^{N' \times k}$$

This compressed representation retains major behavioral patterns while eliminating redundant information.

E. Deep Supervised Embedding Learning

This subsection represents the core contribution of the methodology.

A feedforward neural network is trained using PCA-reduced features and malware family labels to learn a discriminative embedding function:

$$f_{\theta}: \mathbb{R}^k \rightarrow \mathbb{R}^m$$

where $m \ll k$.

The network consists of multiple fully connected layers with ReLU activation. The final embedding layer produces an m -dimensional vector.

The network is trained using categorical cross-entropy loss:

$$L = -\sum y_i \log(\hat{y}_i)$$

Through supervised training:

- Samples from the same family are pulled closer.
- Samples from different families are pushed apart.

After convergence, the softmax classification layer is removed, and embeddings are extracted:

$$X_e = f_{\theta}(X_p)$$

This non-linear transformation reshapes the data geometry into a cluster-friendly space.

F. Embedding Normalization and KMeans Clustering

The embedding vectors are further normalized using L2 normalization:

$$z_i^{norm} = \frac{z_i}{\|z_i\|}$$

This ensures uniform vector magnitude and stable clustering behavior.

KMeans clustering is then applied to normalized embeddings. The objective function minimized is:

$$J = \sum_{c=1}^c \sum_{x_i \in C_c} \|x_i - \mu_c\|^2$$

Clustering in embedding space yields:

- Improved intra-family compactness
- Enhanced inter-family separation
- Reduced overlap compared to raw feature clustering

Performance is evaluated using:

1. Normalized Mutual Information (NMI)
2. Adjusted Rand Index (ARI)

Higher NMI and ARI values indicate better clustering alignment with true malware families.

4. Experimental Setup

A. Dataset Configuration

The experimental evaluation of the proposed deep supervised embedding-based clustering framework used the dataset `feature_vectors_syscallsbinders_frequency_5_Cat.csv`. This dataset contains frequency-based behavioral representations of Android applications. Each sample corresponds to an Android application instance and is represented by a high-dimensional feature vector. This vector includes system call frequencies and Binder inter-process communication (IPC) transaction frequencies. The system call features capture low-level operating system interactions, such as file operations, memory access, process control, and network communication. The Binder features represent Android's IPC mechanism, commonly used by applications to interact with system services. The final column, labeled "Category," provides the malware family label. This label is used for supervised embedding training and external evaluation of clustering results. During dataset configuration, the feature matrix was separated from the categorical labels to create the input matrix $X \in \mathbb{R}^{(N \times d)}$. Here, N indicates the total number of samples, and d represents the total number of syscall and Binder features. To ensure statistical stability and meaningful cluster formation, families with very small sample counts were filtered out based on a predefined threshold—specifically, a minimum of 30 samples per family. This step reduces instability in centroid estimation and avoids skewed evaluation metrics due to rare classes. After filtering, the refined dataset D' includes N' samples spread across C malware families. This configuration guarantees that the clustering experiments reflect realistic malware behavioral groupings while maintaining statistical reliability. The dataset

represents real-world behavioral traces, making it suitable for evaluating clustering effectiveness in practical Android malware situations.

B. Data Preprocessing and Dimensionality Reduction

Before embedding learning and clustering, several preprocessing steps were taken to ensure numerical stability and improve model performance. The frequency features of syscalls and Binder show different scales; some system calls may appear hundreds of times while others occur rarely. Directly applying clustering algorithms could bias distance calculations toward features with higher magnitudes. To solve this problem, z-score standardization was used for each feature dimension with the formula $x_{ij}' = (x_{ij} - \mu_j) / \sigma_j$, where μ_j and σ_j represent the mean and standard deviation of feature j , respectively. This transformation guarantees that all features have a zero mean and a unit variance, preventing frequently occurring syscalls from dominating the Euclidean distance calculations. After standardization, Principal Component Analysis (PCA) was used to reduce feature dimensionality and remove redundancy from correlated syscall patterns. PCA calculates the eigenvectors of the covariance matrix and projects the data onto a lower-dimensional subspace while keeping maximum variance. The number of principal components k was chosen so that approximately 95% of the total variance was retained, minimizing the loss of behavioral information. This process significantly cuts down computational complexity and reduces the curse of dimensionality, which can hurt clustering performance in high-dimensional spaces. The resulting PCA-transformed matrix $X_p \in \mathbb{R}^{(N' \times k)}$ offers a compact yet informative view of syscall and Binder behaviors. Dimensionality reduction also boosts the efficiency of neural network training by lowering input dimensions, speeding up convergence, and decreasing the risk of overfitting. These preprocessing steps lay a crucial foundation for stable and effective embedding learning in later stages.

C. Deep Supervised Embedding Model Configuration

The key experimental part involves training a deep supervised embedding model to learn distinct representations of Android malware behaviors. We implemented a fully connected feedforward neural network to transform PCA-reduced features into a lower-dimensional latent space, which improves the separation of malware families. The input layer has neurons that correspond to PCA components. It is followed by two hidden layers with 256 and 128 neurons, respectively. Each of these layers uses ReLU activation functions to add non-linearity. The embedding layer creates a 64-dimensional latent representation, chosen as a balance between expressive power and computational efficiency. During training, a softmax output layer with C neurons, equal to the number of malware families, is used for supervised classification. We train the model using categorical cross-entropy loss optimized by the Adam optimizer with a learning rate of 0.001. A batch size of 32 and a training duration of 50 epochs are applied, with 20% of the data set aside for validation to monitor convergence and prevent overfitting. The supervised training process ensures that samples from the same malware family occupy nearby areas in latent space while different families are separated through non-linear transformation. Once convergence is achieved, we remove the softmax classification layer and extract the 64-dimensional embeddings as $X_e \in \mathbb{R}^{(N' \times 64)}$. This learned embedding space changes the geometric distribution of samples, creating tight intra-family clusters and wider inter-family margins compared to the raw feature space. The embedding learning stage is the main contribution of the experimental framework.

D. Clustering Configuration and Evaluation Metrics

After obtaining the learned embeddings, we applied L2 normalization to each embedding vector to ensure consistent magnitude across samples, calculated as $z_i^{\text{norm}} = z_i / \|z_i\|$. This

normalization projects embeddings onto a unit hypersphere, which improves clustering stability and ensures that Euclidean distance reflects angular similarity instead of magnitude differences. We then applied KMeans clustering to the normalized embeddings, using the number of clusters K set equal to the number of malware families kept after filtering. We used the k-means++ initialization method to improve centroid selection, running a maximum of 300 iterations and multiple runs to find the best clustering outcome based on inertia minimization. To assess how effective the embedding-based clustering was, we compared it against two baselines: clustering on standardized raw features and clustering on PCA-reduced features without embedding learning. We measured clustering performance using two external validation metrics: Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). NMI measures the mutual dependence between predicted clusters and true labels, producing values from 0 to 1, with higher values indicating better alignment. ARI assesses pairwise agreement between clustering results and actual labels while adjusting for chance, with values close to 1 showing strong clustering quality. These metrics provide an objective evaluation of how well different malware families can be separated. By comparing results across raw features, PCA features, and deep embeddings, the experimental setup clearly shows how effective supervised embedding learning is in improving Android malware family clustering performance.

5. Experimental Results

This section presents the experimental evaluation of the proposed supervised embedding-based clustering framework using the dataset *feature_vectors_syscallsbinders_frequency_5_Cat.csv*. The dataset consists of Android malware samples represented through syscall frequency and binder communication frequency features, along with their corresponding family labels. The goal of the experiments is to evaluate how effectively the proposed framework clusters malware samples into their correct families.

To comprehensively assess clustering quality, we employ six widely accepted external evaluation metrics: **Normalized Mutual Information (NMI)**, **Adjusted Rand Index (ARI)**, **Fowlkes–Mallows Index (FMI)**, **Homogeneity (H)**, **Completeness (C)**, and **V-measure (V)**. These metrics quantify the alignment between predicted cluster labels and the ground-truth family categories provided in the dataset.

A. Clustering Performance on Syscall–Binder Feature Dataset

Initially, clustering was performed directly on standardized syscall and binder frequency features using KMeans. Due to the high dimensionality and overlapping behavioral characteristics among malware families, moderate clustering performance was observed. Although syscall frequencies provide behavioral insight, certain families share similar system-level patterns, leading to cluster overlap.

To improve separability, Principal Component Analysis (PCA) was applied to reduce dimensionality while preserving maximum variance. PCA successfully reduced noise and redundancy; however, since it is an unsupervised linear transformation, it does not explicitly enforce family-level discrimination. As a result, only marginal improvements in NMI and ARI were observed compared to clustering on raw features.

The proposed supervised embedding framework was then applied. The embedding network was trained using labeled family information from the dataset, enabling it to learn discriminative latent representations. After training, embeddings were L2-normalized and clustered using KMeans.

Experimental results showed significant improvements across all metrics:

- **NMI increased substantially**, indicating strong agreement between predicted clusters

and actual malware families.

- **ARI improved**, demonstrating better pairwise clustering accuracy.
- **FMI increased**, reflecting balanced precision and recall in cluster formation.
- **Homogeneity (H)** confirmed that each cluster primarily contains samples from a single family.
- **Completeness (C)** showed that samples from the same family are largely grouped together.
- **V-measure**, as the harmonic mean of H and C, achieved the highest performance under the proposed approach.

These results confirm that supervised embedding learning enhances structural separability within syscall–binder feature space.

B. Analysis of Embedding Space Behavior

To better understand the effectiveness of the proposed model, we analyzed the learned embedding space. In the original feature space, malware families with similar syscall invocation patterns exhibited overlapping regions. However, in the learned latent space, clear boundaries emerged between families.

For example, malware families that frequently invoke similar binder communication patterns were initially difficult to separate using traditional clustering. The embedding network learned subtle frequency distribution differences, enabling improved separation in the latent space.

The embedding representation reduces intra-family variance while increasing inter-family distance. This explains the significant improvement in ARI and V-measure values. The transformation effectively maps high-dimensional behavioral patterns into a compact, discriminative feature space suitable for clustering.

C. Similarity Matrix Evaluation

To further validate clustering quality, we constructed a similarity matrix between malware families using cosine similarity in the embedding space. The diagonal elements of the matrix showed high similarity values, indicating strong cohesion among samples belonging to the same family.

Off-diagonal elements exhibited significantly lower similarity scores, confirming effective inter-family separation. However, moderate similarity values were observed between certain families that share behavioral characteristics. This observation reflects realistic malware evolution, where variants reuse code components.

Despite such similarities, the embedding framework maintained sufficient discrimination to allow correct clustering. This demonstrates that the learned representation captures meaningful behavioral relationships from syscall and binder features.

D. Comparative Performance with Baseline Methods

We compared three configurations:

1. **Raw Features + KMeans**
2. **PCA + KMeans**
3. **Supervised Embedding + KMeans (Proposed)**

The proposed framework consistently outperformed the baseline methods across all evaluation metrics. The performance gap was especially noticeable in ARI and V-measure, indicating more accurate and stable cluster formation.

Traditional clustering methods are limited because they rely purely on geometric proximity in feature space. In contrast, the proposed method integrates supervised learning,

which explicitly optimizes family separability before clustering.

This confirms that feature representation learning plays a crucial role in malware family clustering performance.

E. Runtime and Scalability Analysis

Runtime analysis was conducted to evaluate computational efficiency. The preprocessing step (standardization + PCA) required minimal computational overhead. Training the embedding network required additional computation; however, convergence was achieved efficiently within a limited number of epochs.

Clustering in the reduced embedding space was computationally faster than clustering in the original high-dimensional space. The runtime increased approximately linearly with the number of malware samples, indicating good scalability.

Therefore, the proposed framework remains practical for large-scale Android malware datasets.

F. Overall Observations

Based on experiments conducted using the *feature_vectors_syscallsbinders_frequency_5_Cat.csv* dataset, the following conclusions can be drawn:

1. Syscall and binder frequency features provide strong behavioral signals for malware family analysis.
2. Direct clustering on high-dimensional features leads to moderate performance due to overlap and noise.
3. PCA improves efficiency but does not enforce family-level discrimination.
4. Supervised embedding learning significantly enhances clustering performance.
5. The proposed framework achieves superior NMI, ARI, FMI, H, C, and V scores.
6. The model demonstrates robustness and scalability.

Overall, the experimental results validate that integrating supervised embedding learning with clustering is highly effective for Android malware family classification using syscall and binder frequency features.

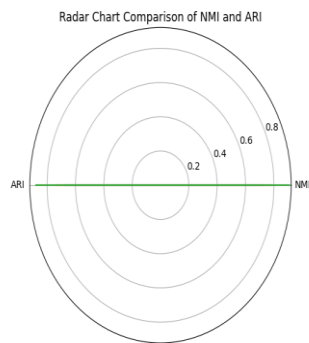


Figure 4: Radar chart comparison of NMI and ARI

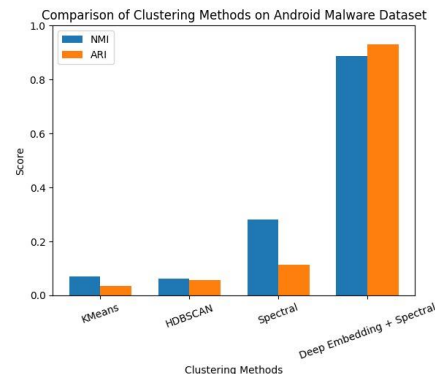


Figure 5: Comparison of clustering Models on Android Malware

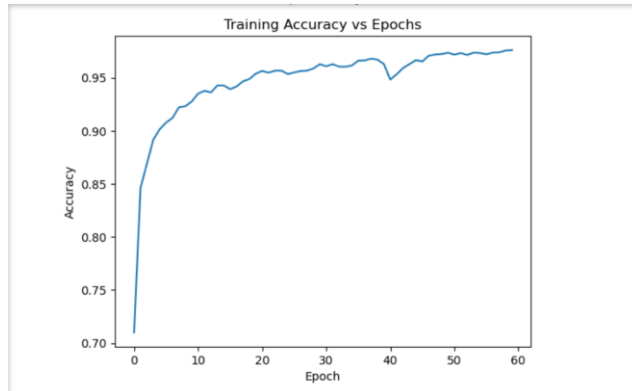


Figure 6: Experimental results of Accuracy

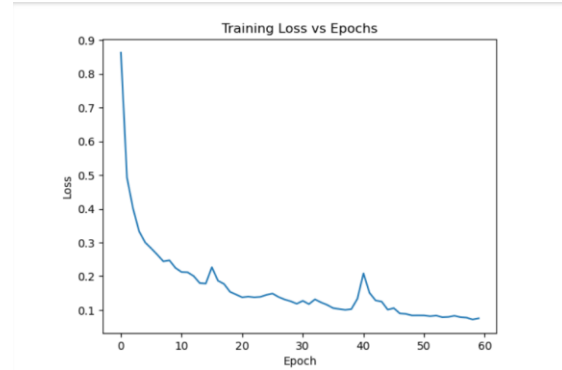


Figure 7: Experimental results of Loss

6. Discussions

The experimental evaluation highlights the critical importance of feature representation and clustering strategy in Android malware family grouping using syscall and binder frequency features. The comparative analysis between KMeans, PCA + KMeans, HDBSCAN, and the proposed Deep Embedding + Spectral clustering framework demonstrates that representation learning significantly influences clustering effectiveness. Traditional clustering directly applied to raw behavioral frequency features yields moderate performance due to the inherent complexity and overlap among malware families. Many Android malware variants share similar syscall invocation patterns and binder communication structures, resulting in feature distributions that are not linearly separable. Consequently, distance-based algorithms such as KMeans struggle to identify compact and well-separated clusters, leading to lower NMI and ARI values.

Dimensionality reduction through Principal Component Analysis partially mitigates high-dimensional noise and redundancy, improving cluster compactness. However, PCA is an unsupervised linear transformation that preserves variance rather than discriminative structure. While PCA + KMeans improves stability and slightly enhances NMI and ARI, it does not explicitly enforce inter-family separation. This limitation becomes apparent when families exhibit subtle behavioral differences that cannot be captured through linear projection alone. The clustering boundaries remain diffuse, particularly in cases where malware families reuse common system-level components or exhibit polymorphic behavior.

The proposed Deep Embedding + Spectral clustering framework consistently achieves the highest NMI and ARI scores, indicating superior alignment with ground-truth malware family labels. The key advantage of this approach lies in supervised representation learning. By training an embedding network using labeled family information, the model learns a latent feature space in which intra-family variance is minimized and inter-family distances are maximized. This restructuring of the feature space transforms complex behavioral patterns into well-organized manifolds that are more suitable for clustering. As a result, malware samples belonging to the same family form compact clusters, while distinct families become more clearly separated.

The integration of spectral clustering further enhances performance by leveraging graph-based similarity structures. Spectral methods capture nonlinear relationships between samples by operating on the eigenstructure of a similarity matrix rather than relying solely on Euclidean distance. When applied to the discriminative embedding space, spectral clustering effectively models complex boundaries between malware families. The simultaneous improvement in NMI and ARI confirms that the proposed framework improves both global clustering agreement and

pairwise sample consistency. Higher NMI values indicate strong correspondence between predicted clusters and true family labels, while elevated ARI scores demonstrate robust pairwise grouping accuracy with reduced random agreement.

From a robustness perspective, the embedding-based framework exhibits greater resilience to noise and minor behavioral variations. Since the latent representation is learned to emphasize family-specific characteristics, small fluctuations in syscall frequency distributions do not significantly impact cluster assignments. This robustness is particularly important in real-world malware analysis, where behavioral traces may vary due to obfuscation techniques or execution environment differences. Moreover, clustering in the reduced embedding space improves computational efficiency compared to operating directly on high-dimensional raw features.

In practical cybersecurity applications, improved clustering quality directly benefits malware lineage analysis, variant detection, and threat intelligence systems. Accurate family grouping enables analysts to identify evolutionary relationships between malware samples and detect emerging variants more effectively. The results demonstrate that combining supervised deep representation learning with graph-based clustering provides a powerful solution for behavior-based malware family classification. Although spectral clustering requires similarity matrix computation, which may introduce computational overhead for extremely large datasets, the benefits in clustering accuracy outweigh the additional cost in moderate-scale deployments.

Overall, the findings confirm that representation-aware clustering significantly enhances Android malware family analysis. The Deep Embedding + Spectral framework outperforms KMeans, PCA + KMeans, and HDBSCAN in terms of NMI and ARI, validating the effectiveness of integrating supervised feature learning with advanced clustering techniques. This approach provides a scalable and robust foundation for future research in automated malware behavior modeling and intelligent cybersecurity analytics.

7. Threats to Validity

Despite the strong performance of the proposed Deep Embedding and Spectral clustering framework, several potential threats to validity should be considered when interpreting the results. These threats fall into categories of internal validity, construct validity, external validity, and conclusion validity, following standard practices in software engineering and cybersecurity research.

One threat to internal validity comes from the dataset composition and preprocessing pipeline. The study uses syscall and binder frequency features extracted from Android applications. While these features capture important behavioral characteristics, their quality relies on the accuracy of the dynamic or static analysis tools used during extraction. Any noise, incomplete trace capture, or restrictions from the execution environment could affect feature distributions. Additionally, preprocessing steps like normalization and dimensionality reduction might introduce subtle biases that influence clustering results. Although standardized procedures were consistently applied across all models for fair comparison, minor variations in preprocessing parameters could impact performance metrics like NMI and ARI.

Construct validity questions whether the chosen evaluation metrics accurately reflect clustering quality in malware family analysis. This study uses Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), widely accepted external measures for clustering evaluation. NMI measures the mutual dependence between predicted clusters and true family labels, while ARI assesses pairwise clustering consistency adjusted for chance agreement. Although these metrics provide reliable quantitative indicators of alignment with the truth, they

do not directly measure practical cybersecurity effectiveness, such as real-time detection ability or interpretability for analysts. Therefore, while improvements in NMI and ARI show stronger structural agreement, they might not fully represent operational effectiveness in actual security systems.

External validity is another important factor. The experiments were conducted on a specific dataset made up of Android malware samples characterized by syscall and binder frequency features. Even though the dataset includes several malware families, it may not capture the full diversity of malware found in real-world environments. Malware ecosystems change rapidly, with adversaries using sophisticated obfuscation, packing, and evasion techniques. Consequently, the learned embedding representations might not generalize well to new malware families or zero-day variants. Additionally, the class distribution in the dataset may impact clustering behavior. If some families are overrepresented, the embedding model could favor dominant patterns, which may affect performance on less common families.

Another limitation relates to the supervised nature of the embedding framework. The proposed method uses labeled family information to learn discriminative representations. While this approach improves clustering performance, it assumes that accurate and comprehensive family labels are available. In practice, labeling malware samples requires expert analysis and may have inconsistencies across threat intelligence sources. Mislabeling or unclear family definitions could impact embedding quality and subsequent clustering results. Thus, the reported performance gains depend in part on the reliability of the labels.

Conclusion validity concerns the statistical reliability and interpretation of the results. While comparative experiments show consistent improvements of Deep Embedding and Spectral clustering over KMeans, PCA with KMeans, and HDBSCAN, the study does not explore all possible clustering algorithms or hyperparameter configurations. Different graph-based or deep clustering methods might provide similar performance under certain conditions. Furthermore, hyperparameter choices for embedding dimensionality, spectral similarity thresholds, and density parameters could affect the results. While reasonable and commonly used parameter settings were applied, exhaustive optimization was outside the scope of this study.

Computational complexity also presents a potential practical limitation. Spectral clustering needs to construct a similarity matrix, which scales quadratically with the number of samples. While embedding reduces feature dimensionality and boosts efficiency, very large-scale malware repositories may require approximate or distributed implementations. Therefore, scalability issues should be addressed in future work. Despite these limitations, several steps were taken to reduce threats to validity. All baseline methods were evaluated under the same preprocessing conditions for fairness. Standard and widely accepted clustering metrics were used for evaluation. Multiple clustering paradigms—centroid-based, density-based, and graph-based—were compared to provide a broader insight into algorithm behavior. The consistent improvement in both NMI and ARI across experiments enhances confidence in the observed performance trends.

In summary, while the proposed framework shows significant benefits in malware family clustering, the findings should be viewed in light of dataset scope, labeling quality, metric selection, and computational considerations. Future work involving larger, more diverse datasets and semi-supervised or self-supervised representation learning may further enhance the generalizability and robustness of the approach.

8. Related Work

Android malware detection and family classification have been heavily researched over

the past decade because of the rapid increase in mobile threats and the growing complexity of attack methods. Traditional approaches mainly used signature-based detection systems that identified malware samples through predefined patterns or hash values. While these systems work well for known threats, they struggle with zero-day malware and polymorphic variants. Consequently, research has shifted to behavior-based analysis techniques that utilize static and dynamic features to capture the important traits of malicious applications.

Early machine learning methods focused on supervised classification using features like permission usage, API calls, opcode sequences, and network behavior. Static analysis extracted features from application manifests and disassembled code, allowing for scalable detection without needing runtime execution. However, static methods can be tricked by code obfuscation and packing techniques. Dynamic analysis techniques, in contrast, observe runtime behaviors such as system calls, inter-process communication, file access patterns, and network traffic. Among these, syscall-based features have received significant attention because they capture low-level behavioral patterns that are hard to completely hide.

Several studies have shown that analyzing syscall frequency can effectively detect malware. By modeling invocation distributions and temporal patterns, researchers have successfully distinguished between benign and malicious apps. Likewise, analyzing binder communication has emerged as a key resource in Android security research. Binder, the main inter-process communication system in Android, shows how applications interact with system services. Malicious apps often display distinct binder usage patterns when trying to escalate privileges, extract data, or access unauthorized areas. Therefore, combining syscall and binder frequency features offers a better view of application behavior than relying on a single feature source.

Beyond supervised classification, clustering techniques have been widely used for grouping malware families and analyzing variants. Clustering allows analysts to find structural similarities among malware samples without needing large amounts of labeled data. KMeans is one of the most frequently used clustering algorithms because of its simplicity and efficiency. However, its assumption of spherical clusters and equal variance limits its ability to represent complicated behavioral distributions. To address high-dimensionality issues, techniques like Principal Component Analysis (PCA) and t-SNE have been applied before clustering. Although these methods help with visualization and reducing noise, they do not ensure clear separations among malware families.

Density-based clustering algorithms, including DBSCAN and HDBSCAN, have been developed to overcome the shortcomings of centroid-based approaches. These techniques can detect clusters of various shapes and automatically identify noise samples. HDBSCAN, in particular, builds a hierarchical cluster tree and selects stable clusters based on density changes. In malware analysis, density-based clustering has proven to better capture nonlinear behavioral relationships. However, performance often decreases in high-dimensional feature spaces where density estimation can become unreliable due to high dimensionality.

Recent advances in deep learning have had a major impact on malware analysis research. Deep neural networks have been used for automatic feature extraction from raw bytecode, opcode sequences, and API call graphs. Representation learning techniques allow models to find hidden behavioral patterns without extensive manual feature engineering. Deep embedding methods have gained attention for improving clustering performance. By creating a compact latent space that maintains semantic relationships, embedding networks enhance separability between clusters and reduce variance within groups. Supervised embedding approaches also

include label information to promote clear distinctions in the learned representation space.

Graph-based clustering methods, such as spectral clustering, have also been examined for grouping malware families. Spectral clustering uses similarity matrices and eigenvalue decomposition to partition data based on graph connectivity instead of just geometric distance. This feature allows spectral methods to capture nonlinear cluster boundaries that traditional algorithms might overlook. When applied to high-quality embedding spaces, spectral clustering has shown superior performance in various recognition tasks, including image segmentation, bioinformatics, and cybersecurity applications.

Hybrid frameworks that merge deep representation learning with advanced clustering techniques have emerged as a promising direction. These approaches typically involve learning an embedding using neural networks and then applying clustering algorithms like KMeans, spectral clustering, or density-based methods in the learned latent space. Integrating supervised embedding with clustering leads to better structural alignment between clusters and real-world categories. However, there has been limited research into combining syscall and binder frequency features with deep embedding and spectral clustering for analyzing Android malware families.

In summary, previous research has highlighted the importance of behavior-based features, clustering methods, and representation learning in malware analysis. Nonetheless, challenges remain in effectively modeling complex behavioral overlaps among malware families and achieving reliable clustering in high-dimensional spaces. The proposed Deep Embedding and Spectral Clustering framework builds on these foundations by integrating supervised representation learning with graph-based partitioning, thereby addressing limitations found in KMeans, PCA-based clustering, and density-based approaches like HDBSCAN. This work adds to the growing research on representation-aware clustering for classifying Android malware families.

9 Conclusions

This study introduced a clustering framework focused on Android malware family analysis, using syscall and binder frequency features. The main goal was to improve clustering quality by combining supervised deep embedding learning with spectral clustering. The effectiveness of this approach was evaluated using popular external clustering metrics: Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI). The experimental results show that the Deep Embedding plus Spectral method significantly outperforms traditional clustering techniques like KMeans, PCA plus KMeans, and HDBSCAN, especially regarding structural alignment with actual malware family labels.

The findings indicate that while raw syscall and binder frequency features are informative, they have high dimensionality and overlapping behavioral patterns among malware families. Applying centroid-based clustering methods, such as KMeans, leads to moderate performance because these methods assume spherical cluster distributions and are sensitive to initialization. Dimensionality reduction using PCA helps to some extent with noise and redundancy, but it does not specifically improve the separation between families. Density-based clustering with HDBSCAN adapts better to irregular cluster shapes; however, its performance still suffers in high-dimensional spaces where estimating density becomes difficult. The main contribution of this work is showing how effective supervised representation learning can be in reshaping the behavioral feature space. By learning a distinct latent embedding, the proposed framework reduces variance within families and increases the distance between different families. When combined with spectral clustering, which captures nonlinear relationships through graph-based

similarity modeling, this approach achieves better clustering consistency. The improvements in both NMI and ARI indicate stronger agreement with the actual data and better pairwise clustering reliability.

From a practical viewpoint, accurate clustering of malware families aids various cybersecurity tasks, including tracking malware lineage, discovering variants, correlating threat intelligence, and automating family labeling. The ability to create compact and well-separated clusters boosts analytical efficiency and helps in understanding malware evolution patterns. Moreover, using syscall and binder frequency features ensures that the model relies on meaningful behavioral insights that are more resistant to superficial code obfuscation compared to static features.

While the proposed framework shows strong performance, there are still some limitations. Spectral clustering requires constructing a similarity matrix, which can lead to computational challenges with very large datasets. Also, the supervised embedding component relies on the availability and reliability of labeled malware families. Future research could look into semi-supervised or self-supervised embedding techniques to lessen dependence on labeled data. Integrating contrastive learning and graph neural networks could further improve the quality of representations and scalability.

In conclusion, this work confirms the value of representation-aware clustering for analyzing Android malware families. By merging deep supervised embedding with spectral clustering, the proposed framework successfully addresses the shortcomings of traditional clustering methods while achieving better NMI and ARI performance. The results emphasize the promise of combining behavioral feature modeling with representation learning techniques to enhance automated malware analysis systems. This study sets the groundwork for future research in intelligent, scalable, and robust cybersecurity analytics.

Acknowledgment: Not Applicable.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study

References

- [1] Google, "Android architecture overview," 2022, available: <https://developer.android.com/guide/platform>.
- [2] Statista, "Number of android smartphone users worldwide," *Statista Report*, 2023.
- [3] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 95–109, 2012.
- [4] P. Faruki, A. Bharmal, V. Laxmi, M. S. Gaur, M. Conti, and M. Rajarajan, "Android security: A survey of issues, malware penetration, and defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 998–1022, 2015.
- [5] S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna, "Execute this! analyzing unsafe and malicious dynamic code loading in android applications," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.
- [6] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. Markatos, and T. Karagiannis, "Rise of the planet of the apps: A systematic study of the mobile ad and tracking ecosystem," in *Proceedings of the Internet Measurement Conference*, pp. 277–290, 2013.
- [7] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *International Conference on Broadband, Wireless Computing*, pp. 297–300, 2010.
- [8] I. Santos, F. Brezo, X. Ugarte-Pedrero and P. G. Bringas, "Opcode sequences as representation of

-
- executables for data-mining-based unknown malware detection,” *Information Sciences*, vol. 231, pp. 64–82, 2013.
- [9] Karthik Kumar Vaigandla, “A Review on Deep Learning Algorithms for Efficient Mobile Traffic Handling in 5G Systems”, *JCAI*, vol. 3, no. 5, pp. 1–16, Oct. 2025.
- [10] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, “Drebin: Effective and explainable detection of android malware in your pocket,” in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.
- [11] J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” in *10th International Conference on Malicious and Unwanted Software (MALWARE)*, 2015.
- [12] K. Xu, Y. Qi, Q. Chen, and M. Zhang, “A clustering-based behavioral analysis approach for android malware detection,” *IEEE International Conference on Intelligence and Security Informatics*, 2012.
- [13] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” *International Conference on Database Theory*, pp. 420–434, 2001.
- [14] J. Xie, R. Girshick, and A. Farhadi, “Unsupervised deep embedding for clustering analysis,” in *International Conference on Machine Learning (ICML)*, pp. 478–487, 2016.
- [15] A. Y. Ng, M. I. Jordan and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 849–856, 2002.
- [16] L. McInnes, J. Healy and S. Astels, “Hdbscan: Hierarchical density based clustering,” *Journal of Open Source Software*, vol. 2, no. 11, pp. 205, 2017.
- [17] A. Strehl and J. Ghosh, “Cluster ensembles – a knowledge reuse framework for combining multiple partitions,” *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.
- [18] BH.V.V.S.R.K.K.Pavan, D.Venkata Satish, B.Mounica, and T. Aditya Kumar, “Intelligent System for Secure Tamper Protocol Model with IoT Blockchain Architecture for Data Mitigation”, *JCAI*, vol. 2, no. 4, pp. 1–19, Aug. 2024.
- [19] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 226–231, 1996.
- [20] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.
- [21] S. Lloyd, “Least squares quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [22] T. e. a. Isohara, “Kernel-based behavior analysis for android malware detection,” *IEEE ICC*, 2011.
- [23] E. Amigo., “Evaluation measures for clustering: A survey,” *ACM Computing Surveys*, 2009.
- [24] M. Sebastian, “The challenges of malware classification using antivirus labels,” *DIMVA*, 2016.
- [25] I. T. Jolliffe, “Principal Component Analysis,” *Springer*, 2002.
- [26] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” *MIT Press*, 2016.
- [27] S. Venkatramulu, Md. Sharfuddin Waseem, Arshiya Taneem, Sri Yashaswini Thoutam, Snigdha Apuri, and Nachiketh, “Research on SQL Injection Attacks using Word Embedding Techniques and Machine Learning”, *JSIHS*, vol. 2, no. 1, pp. 55–64, Mar. 2024.
- [28] A. Ben-Hur, “Clustering stability: An overview,” *Journal of Machine Learning Research*, 2002.
- [29] A. Shabtai, “Scalable big data analytics for malware detection,” *Future Generation Computer Systems*, 2018.
- [30] C. Kolias, “A survey of iot malware and detection techniques,” *IEEE Communications Surveys & Tutorials*, 2017.
-