An ML-Driven Approach for Detecting Malicious PDFs within an Android Security Advisor

1st Esha Fatima
Faculty of Engineering and Technology
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
eshafatima394@gmail.com

2nd Areesha
Faculty of Engineering and Technology
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
ejazareesha4@gmail.com

3rd Tehreem Amna
Faculty of Engineering and Technology
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
tehreemamna005@gmail.com

4th Asjad Amin
Faculty of Engineering and Technology
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
asjad.amin@iub.edu.pk

5th Umar Fayyaz
Faculty of Engineering and Technology
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
umar.fayyaz@iub.edu.pk

***Abstract*—Android smartphones have been a popular means for people to connect to digital services and hence a lucrative target for malicious attacks. The Portable Document Format has also been a popular means of sharing files on Android platforms and has also started to be used increasingly for malicious attacks through inbuilt scripts, objects, and obfuscation methods in PDF files. Signature-based methods have been inefficient in defending against modern and novel attacks. A machine learning-based approach for malicious PDF file detection in an Android Security Advisor application is described in this paper. The developed system extracts structural and metadata-based features from PDF files and utilizes a neural network-based model trained offline and implemented on Android smartphones via TensorFlow Lite. The detection framework preserves parity between training and mobile-based data preprocessing via metadata-based normalization. The experimental outcome indicates that the developed technique is highly efficient in malicious PDF file detection and has a lower false positive ratio and is optimized for Android-based platforms.**

***Index Terms*—Android Security, Malicious PDF Detection, Machine Learning, Mobile Malware Analysis, Machine Learning, Deep Learning, TensorFlow Lite, Structural Feature Extraction.**

I. Introduction

Android is currently the most widely used operating system. More than 2.5 billion people actively use it every month [1] [2]. Since most users access the internet and digital services through mobile devices, Android has become the most common platform globally. Its use case is expanding day by day, including communication, video streaming, entertainment, banking apps, and controlling various devices. Many of these areas are sensitive because of security and privacy. Because of this, Android as an operating system needs to provide strong security so that users can trust it. PDF (Portable Document Format) files are one of the most widely used document types for sharing information across different digital platforms, especially on Android devices. They are popular because their layout is easy to view and the format is simple for anyone to use. The widespread use also increased the attackers interest in targeting PDFs. Attackers often embed exploits, malware, or phishing payloads within PDF files. The risk is high on Android devices due to their open nature and ecosystem. Traditional methods such as signature-based antivirus rely on signature matching, which means they can only detect malware that is already known and stored in their database. Attackers now use advanced techniques that easily bypass traditional methods [3]. This method is no longer enough to handle modern attacks or obfuscated malware that attackers commonly hide inside PDF files, especially on android devices. Because Android is the most widely used platform, and PDFs are one of the most commonly shared document types through Whatsapp, Telegram and web browsers , attackers target them frequently. Machine learning has become an effective method for malware detection because it can learn behaviour patterns from data. Machine learning models can identify new and obfuscated malware by learning generalized patterns [4].This makes machine learning detection more suitable for modern attacks including malicious PDFs, that often hide harmful content inside complex structures.

Identify applicable funding agency here. If none, delete this.

II. System Architecture and working

The proposed system detects malicious or insecure PDFs employing a machine learning-based approach [5]. The system architecture includes several interconnected modules begin-

DOI: https://doi.org/10.67535/tsp.000002.001
Published under the ISAISS 2026 Proceedings

ning with dataset creation and feature extraction followed by model training finally deployment into the backend that communicates with the Android application, each performing one specific task in the detection pipeline. The architecture of the system allows for efficient processing of PDF files, right from feature extraction to classification, with very little human interaction.

The process is started by collecting benign and malicious PDF files, followed by the extraction of structural features [6]. such as object counts, metadata size, JavaScript indicators, encryption flags, and header versions as shown in 1. These features form the basis for training a binary classification model. Once the model is trained, it is converted to TensorFlow Lite format for mobile deployment. A metadata file containing feature order and normalization parameters ensures consistency between training-time pre processing and the Android-side.

On Android, the user selects a PDF file, after which its Features are extracted, standardized, and passed to the TFLite [7] interpreter. The model outputs a probability score that indicates whether the document is malicious or benign. The user interface then displays the final result.

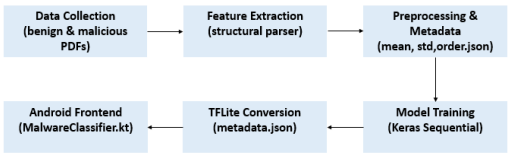

Fig. 1. System architecture of the ML-based PDF detection system, showing the workflow from data collection to on-device inference.

III. DATASET AND FEATURE ENGINEERING

The dataset consists of malicious and benign PDF files collected from publicly available repositories and search archives. Malicious samples include PDF containing JavaScript exploits, embedded objects, encrypted streams, or obfuscated payloads. Benign files represent common educational and official documents free of executable content. We used lightweight Structural features that are inexpensive to compute and robust to simple obfuscations and lightweight enough for mobile execution [8]. Table I lists the main features and their descriptions. A PDF parser in Python extracts the raw counts and indexes by scanning the PDF tokens with a tolerant lexer. For on-device use, an equivalent Kotlin-based extractor provides the same raw features.

These selected structural features [9] are summarized in Table I. The **PDF size**, which can increase for malicious files with packed streams or embedded payloads; **metadata size**, as manipulated or suspiciously large metadata blocks can indicate tampering; **page count**, because very low or suspiciously high counts have been correlated with obfuscated malicious documents; **xref table length**, which often increases when attackers insert hidden objects; **and the length of the title/author in characters**, which can reflect auto-generated or otherwise

TABLE I
SELECTED STRUCTURAL FEATURES

Feature	Description
pdfsize	File size in KB
metadata size	Bytes of PDF metadata
pages	Number of pages
xref Length	Length of cross-reference table
title characters	Number of title characters in metadata
isEncrypted	Binary flag indicating whether file is encrypted
embedded files	Count of embedded files
images	Total image objects
obj, endobj	Counts of object markers
stream, endstream	Counts of stream markers
xref, trailer, startxref	Structural markers occurrences
pageno	Page number tokens
ObjStm	Object stream occurrences
JS, Javascript	Presence of JavaScript tags
OpenAction, Acroform	Action/form indicators
RichMedia, EmbeddedFile	Embedded resource indicators
header_x.x	One-hot encoded PDF header version

suspicious document creation.The presence of **JavaScript tags** (JS, JavaScript) is a strong signal, as exploit-based malware frequently embeds hidden scripts to trigger payload execution,count of **embedded files** and total embedded resources often increases in malicious samples because attackers attach executables, shellcode, or decoy files inside the PDF container,number of **image objects** and **total image streams** may rise when adversaries use images or image streams to conceal encrypted payloads,**xref, trailer, and startxref** are also monitored, as unusual frequencies may indicate reconstruction of the PDF object graph or tampering with object references-some common obfuscation techniques.And rare PDF tokens to further help discriminate malicious manipulation of normal document structure.A Kotlin-based extractor provides the same feature set on-device for the Android classifier [10].

In this work, the maliciousness probability produced by our trained TensorFlow Lite model is the target feature. It assigns to the extracted structural features a single output value between 0 and 1, whose values close to 0 indicate a benign PDF, while values closer to 1 mean a high likelihood of malicious behavior. Thus, possible outcomes of the model are: (1) a benign prediction, where the document's structural patterns match normal PDF behavior, and (2) a malicious one, since the document exhibits characteristics found in malware-infected PDFs such as heavily obfuscated streams, suspicious JavaScript, abnormal metadata patterns, or injected objects. This single-probability output is lightweight, interpretable, and suitable for real-time malware detection on mobile devices.

IV. MACHINE LEARNING MODEL

A Keras-based Sequential model was trained using the standardized features. The architecture consisted of dense layers ReLU activation, dropout layers for regularization, and the sigmoid output layer for binary classification. The model was trained using the Adam optimizer and binary cross-entropy loss.

After training, the model was converted into TFLite format. to enable the execution on Android devices. Along with the model, file, the feature metadata JSON ensures that mobile-side Preprocessing aligns with the training environment.

During offline training the following operations are applied: to each numeric feature:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

where μ and σ denote the mean and standard deviation computed on the training data. These values, together with the exact order of the features, are serialized into 'metadata.json'. A small The JSON excerpt is abbreviated, as seen in Listing 1.

"feature$_order$" : [
"$pdfsize$",
"$metadatasize$",
"$pages$",
"$xrefLength$",
"$titlecharacters$"...
],

Keras Sequential Model The model used during training is a compact feed-forward network to balance performance and inference cost [11]. Table II describes the architecture.

TABLE II
KERAS SEQUENTIAL MODEL ARCHITECTURE (EXAMPLE)

Layer	Units	Activation / Notes
Input	42	Feature vector
Dense	128	ReLU
Dropout	0.3	Regularization
Dense	64	ReLU
Dropout	0.2	Regularization
Dense	32	ReLU
Output	2	Softmax (or single sigmoid)

The model is trained with Adam optimizer, learning rate, with early stopping monitored on validation loss. Loss: binary cross-entropy, or categorical cross-entropy for 2-output softmax). Standard training/validation/test split (e.g 70/15/15) is used.

Baseline comparisons include Random Forest and Logistic Regression [12]. These baselines serve to validate whether the Neural model considerably improves the detection metrics [13].

An important consideration is that the on-device preprocessing is identical to the offline training. We conducted three key Kotlin components:

- **ModelMetadata.kt:** Kotlin data classes to deserialize 'metadata.json'.
- **AssetLoader.kt:** Loads and parses JSON, provides scaling helper methods.
- **MalwareClassifier.kt:**Loads the TFLite model into Interpreter, Handles ByteBuffer conversion and inference.
  The model outputs a probability distribution. Example: output '[0.30, 0.70]' implies 70 percent probability of malicious. The application applies a threshold (e.g., 0.5) or configured one. thresholds depending on desired trade-off between false positives and false negatives. In sensitive/critical contexts a higher Threshold can be utilized to reduce false alarms.

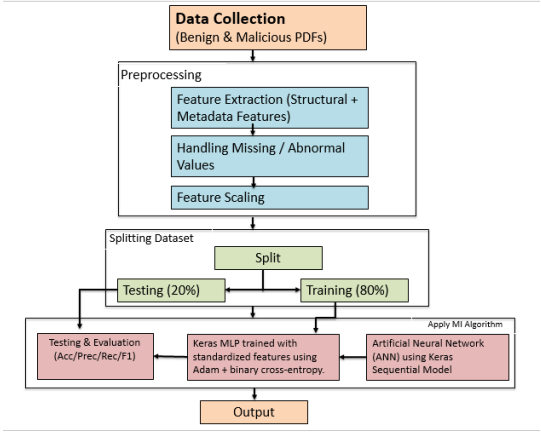

Fig. 2. Workflow of Machine learning Model

The proposed system follows a structured pipeline for the detection of malicious PDFs using machine-learning techniques [14].The workflow 2 broadly consists of five major stages: data collection, preprocessing, dataset splitting, model training, and testing and evaluation.

1) Data Collection The process initiates by acquiring a labeled dataset that includes both benign and malicious PDF files. These files serve as the primary input for extracting discriminative features to enable machine learning–based classification.
2) Preprocessing In this stage, the raw PDF files are transformed into meaningful numerical representations suitable for machine learning models. Preprocessing is further divided into:
   Feature Extraction: Structural and metadata-based features are extracted such as embedded objects, JavaScript tags, streams, size of metadata, number of pages from each PDF. These features capture typical patterns present in malicious files.
   Handling Missing/Abnormal Values: Any missing or inconsistent feature values are identified and corrected to ensure dataset reliability.
   Feature Scaling: The pre-processing step where numerical features are standardized so that each feature contributes equally to the learning process and avoids bias due to different scales [15].
3) Splitting the Dataset The pre-processed dataset is then divided into two sub-sets:
   Training Set: 80
   Testing Set (20
   This split ensures that the model's performance reflects real-world performance in the evaluation metrics.
4) Model Training Two machine learning models are trained on the standardized feature set:

Keras-based Multilayer Perceptron [16]: The model was trained with the Adam optimizer and binary cross-entropy loss.
Artificial Neural Network ANN by Keras Sequential Model: The network learns to classify the PDFs into benign and malicious, depending on the pattern extracted.
Both models are designed to capture nonlinear relationships and high-level representations within the feature set [17].

5) Testing and evaluation
The model trained is evaluated against the test dataset of 20%. The performance is measured using standard metrics for classification [18]:
Accuracy
Precision
Recall
F1-Score
Together, these metrics determine the effectiveness of the system in accurately identifying malicious PDFs and minimizing false alarms.
6) Output
The output will be an individual PDF document classified either as benign or malicious, along with the evaluation metrics [19].

V. Performance Analysis

To assess the efficiency of the proposed PDF malware detection model, a comprehensive performance assessment was carried out using the standard metrics applied in binary classification systems for machine learning. Since the task is to classify malicious PDFs from benign ones, the model outputs are probability scores in the range of [0,1] [20] where values closer to 1 indicate higher malicious likelihood. The predicted class is obtained by applying a decision threshold of 0.5. Performance was measured on a held-out test set not seen during training. The key evaluation parameters are Accuracy, Precision, Recall, F1-Score, False Positive Rate (FPR), and Mean Squared Error (MSE). Each of these metrics quantifies complementary aspects of the quality of detection: to correctly flag malicious files, avoid mislabeling benign ones, and overall robustness. The following mathematical definitions were used:

A. Performance Metrics

The performance of the PDF malware detection model is evaluated using standard classification metrics. The mathematical definitions are given below.

B. Performance Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1\text{-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

$$\text{Mean Squared Error (MSE)} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Where TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively.

TABLE III
Performance Metrics of the Malware Detection Model

Metric	Value
Accuracy	0.963
Precision	0.955
Recall	0.972
F1-Score	0.963
False Positive Rate (FPR)	0.041
Mean Squared Error (MSE)	0.028

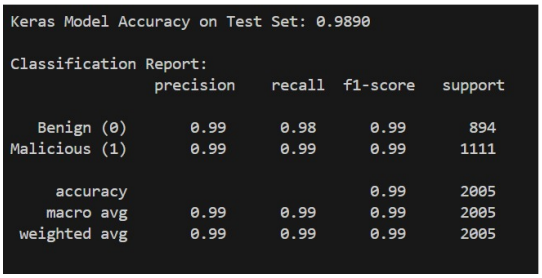

Fig. 3. Representative Evaluation Results

The results in fig 3 confirm that the proposed model based on lightweight structural features classifies malicious files from benign documents reliably and can be applied to resource-constrained environments like mobile devices [21].

VI. Conclusion

The surge in the adoption rate of Android smartphones, as well as the growing use of such devices in sensitive security tasks, has widened the attack surface against mobile malicious software [22]. Specifically, PDF documents continue to serve as a very popular carrier of malicious payloads because of the complexity inherent in such documents. The classical malware analysis technique is not capable of countering modern malware threats, particularly when zero-day attacks are considered.
In this research, we introduced a framework with ML capabilities that can identify malicious PDFs in the Android

Security Advisor [23]. Our solution utilizes a structural-level feature set that is efficient, together with a neural network design and implementation in Keras, optimized with TensorFlow Lite [24]. The design is such that offline learning is consistent with mobile preprocessing, thanks to metadata-driven normalization.

From the experimental findings, it is evident that the proposed solution is capable of identifying malicious PDFs with high accuracy while ensuring that the false positives remain low. In future research, experiments with the incorporation of dynamic behavior-based attributes, more sophisticated deep learning models, and dynamic threshold mechanisms may be considered to further optimize malicious PDF detection [25]. Privacy-preserving solutions such as federated learning can also be adopted within the solution to ensure that the models generalize better while safeguarding user privacy.

REFERENCES

[1] R. Mayrhofer, J. V. Stoep, C. Brubaker, and N. Kralevich, "The android platform security model," *ACM Transactions on Privacy and Security (TOPS)*, no. 3, pp. 1–35, 2021.

[2] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124 579–124 607, 2020.

[3] R. Martin, R. Pava, and S. Mishra, "Analyzing machine learning algorithms for antivirus applications: a study on decision trees, support vector machines, and neural networks." *Issues in Information Systems*, vol. 25, no. 4, 2024.

[4] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," *Computers & Security*, vol. 137, p. 103582, 2024.

[5] K. Mahmud Sujon, R. Binti Hassan, M. Abdullah-Al-Wadud, and J. Uddin, "Optistack: A hybrid ensemble learning and xai-based approach for malware detection in compressed files," *IEEE Access*, vol. 13, pp. 104 992–105 026, 2025.

[6] Various, "Malware classification using machine learning and deep learning: A comprehensive approach," *Cureus Journal*, 2024.

[7] A. Ignatov, G. Malivenko, and R. Timofte, "Fast and accurate quantized camera scene detection on smartphones, mobile ai 2021 challenge: Report," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 2558–2568.

[8] R. Gutierrez, W. Villegas-Ch, L. Naranjo Godoy, A. Mera-Navarrete, and S. Luján-Mora, "Application of deep learning models for real-time automatic malware detection," *IEEE Access*, vol. 12, pp. 107 742–107 756, 2024.

[9] T. M. Mohammed, L. Nataraj, S. Chikkagoudar, S. Chandrasekaran, and B. Manjunath, "Hapssa: Holistic approach to pdf malware detection using signal and statistical analysis," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, 2021, pp. 709–714.

[10] S. Ahmed and M. Khan, "Permission-based android malware detection using ensemble learning," *IEEE Transactions on Mobile Computing*, 2023.

[11] S. Chen *et al.*, "Deep learning based android malware detection using real devices," *Computers & Security*, vol. 104, p. 102096, 2021.

[12] H. Bae, Y. Lee, Y. Kim, U. Hwang, S. Yoon, and Y. Paek, "Learn2evade: Learning-based generative model for evading pdf malware classifiers," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 4, pp. 299–313, 2021.

[13] I.-L. Lin, H.-C. Yang, G.-L. Gu, and A. Lin, "A study of information and communication security forensic technology capability in taiwan," in *IEEE 37th Annual 2003 International Carnahan Conference onSecurity Technology, 2003. Proceedings.*, 2003, pp. 386–393.

[14] I. Almomani *et al.*, "Automated machine learning for deep learning based malware detection," *Computers & Security*, vol. 134, p. 103482, 2023.

[15] S. Y. Yerima, A. Bashar, and G. Latif, "Malicious pdf detection based on machine learning with enhanced feature set," in *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2022, pp. 486–491.

[16] M. AlShoulie *et al.*, "Deep learning approaches for malware detection," *IEEE Access*, 2025.

[17] M. Torres, R. Álvarez, and M. Cazorla, "A malware detection approach based on feature engineering and behavior analysis," *IEEE Access*, vol. 11, pp. 105 355–105 367, 2023.

[18] S. Kumar and B. Gupta, "Hybrid cnn-random forest framework for elevated accuracy in powdery mildew disease classification," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, 2024, pp. 1–6.

[19] J. Zhang *et al.*, "Mlpdf: An effective machine learning based approach for pdf malware detection," *arXiv preprint arXiv:1808.06991*, 2018, cited by 46.

[20] N. T. Cam *et al.*, "uitpdf-malde: Malicious portable document format files detection using multi machine learning models," *Engineering Applications of Artificial Intelligence*, vol. 129, p. 105678, 2025.

[21] M. Issakhani *et al.*, "Pdf malware detection based on stacking learning," in *Proc. Int. Conf. Information Systems Security Privacy*. SCITEPRESS, 2022, pp. 561–570.

[22] A. Alfaw, M. Rouached, and A. Akremi, "A resilient deep learning framework for mobile malware detection: From architecture to deployment," *Future Internet*, vol. 17, no. 12, 2025. [Online]. Available: https://www.mdpi.com/1999-5903/17/12/532

[23] D. Maiorca, B. Biggio, and G. Giacinto, "Towards adversarial malware detection: Lessons learned from pdf-based attacks," *ACM Comput. Surv.*, vol. 52, no. 4, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3332184

[24] B. Team *et al.*, "Lightweight on-device detection of android malware based on machine learning," *Sensors*, vol. 22, no. 17, p. 6612, 2022.

[25] V. Mai, N. Srndic, and R. Sommer, "Evasive-pdfmal2022: An evasion dataset for pdf malware detection," *arXiv preprint arXiv:2302.01389*, 2023.