# MAL-DROID: ANDROID MALWARE DETECTION WITH DEEP NEURAL MODELS

**Guided By:** Dr. Bharati Mishra


**Presented By:**

Nitin Chandra Sahu (B521039)

Harshit Goel (B421023)

Shradha Suman Pal (B521055)

# Contents

# Literature Survey

Android's rapid growth, with over 3 billion users by 2022 **[13]**, makes it a prime malware target, necessitating advanced detection. CNNs, effective in image recognition, excel in Android malware detection using image-based app data representations.

## Image-Based Malware Detection:

- **Nataraj et al. (2011):** Malware binaries as grayscale images, high accuracy, low computational cost **[1].**
- **Foundation for CNN-**based Android malware detection using APK elements as images.

## CNN-Based Detection Techniques:

- **Ghandour et al.:** Permissions from AndroidManifest.xml to images, 93% accuracy (Drebin, APKMirror) **[2].**
- **Almomani & Khayer:** APK components (Manifest, DEX, resources.arsc) to grayscale images, 98.75% accuracy (Drebin, Malgenom, Google Play) **[3].**
- **Kumar et al.:** Full APKs to grayscale, 96.86% accuracy (AMD, Drebin) **[4].**
- **Advanced Methods:**
  - **IMCFN:** App binaries to 2D color arrays, 98.82% accuracy (Maling) **[5].**
  - **R2-D2:** DEX bytecode to RGB images, 99% accuracy (2017 dataset) **[6].**

# Literature Survey

## CNN-LSTM and Hybrid Models:

- **Aldini & Petrelli:** CNN-LSTM on APK elements (256x256 grayscale), 99.18% accuracy, 98.59% precision, 99.19% F1-score (AndroZoo, Drebin) [base paper].
- **AdMat:** CNN-LSTM, 98.26% detection accuracy, 97% classification accuracy **(Drebin, AMD) [7].**
- **CNN-SVM hybrids:** 100% binary accuracy on MalNet images **[8].**

## Dataset Usage and Evaluation:

- **Common datasets:** Drebin (5,560 malware, 179 families) [5][16], AndroZoo (benign apps) **[11]**, AMD (1,000–20,000 samples).
- **Metrics:** Accuracy, precision, recall, F1-score; best results >98%.
- **Example:** DroidMalwareDetector, 90% accuracy (14,000-app dataset) **[12]**.

## Conclusion:

- **CNN-based methods achieve >95% accuracy; CNN-LSTM excels in spatial & sequential analysis.**
- **Future challenges:** Scalability and anti-evasion techniques.

# Problem Statement

**Problem Addressed :** Correct identification of Android malware using deep learning models.

## Objective :

- Develop a static analysis system for Android malware detection using deep learning.
- Leverage CNNs and CNN-LSTM models to automatically extract features from APK files.
- Achieve high accuracy, precision, recall, and F1-scores on benchmark datasets.

## Solution Approach :

- Developed an end-to-end automated malware detection system using CNN and CNN-LSTM, eliminating the need for manual feature engineering.
- Achieved high detection accuracy, outperforming traditional methods in identifying Android malware.

# Solution Approach

## Methodology Diagram :

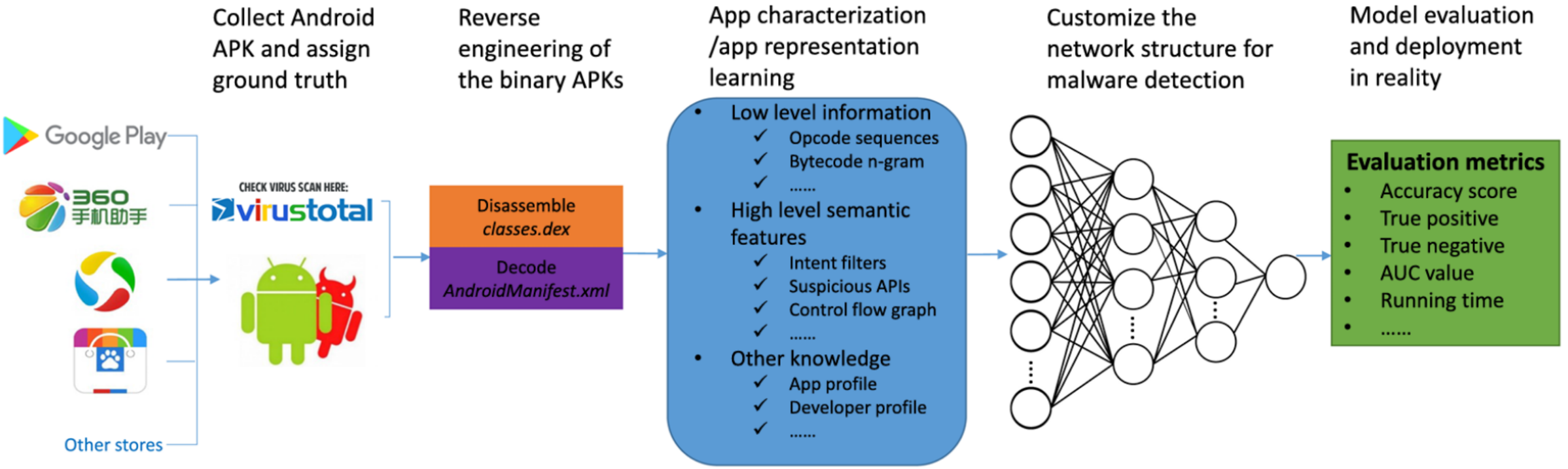



Fig: Five-step methodology for static malware detection [9]

# Data Flow

**Third Step**

Characterization of APKs

**Second Step**

Reverse Engineering (Feature extraction)

**Fourth Step**

Neural Model Adaptation (CNN/CNN-LSTM based classification)

**First Step**

Data Collection (AndroZoo, Drebin dataset)

**Fifth Step**

Model Evaluation

# IMPLEMENTATION DETAIL

# 1. Data Collection and preprocessing of APKs

**Dataset Used**

- Size: 5,000 APK files
- Composition:
  - 2,500 benign apps from AndroZoo.
  - 2,500 malware apps from Drebin.
  - Benign App Criteria: Selected from Google Play store via AndroZoo.
  - Malign App Criteria: Selected from different Malware classifications via Drebin.
- Both datasets are split 80/20:
  - Training Set: 4,000 samples
  - Validation/Test Set: 1,000 samples

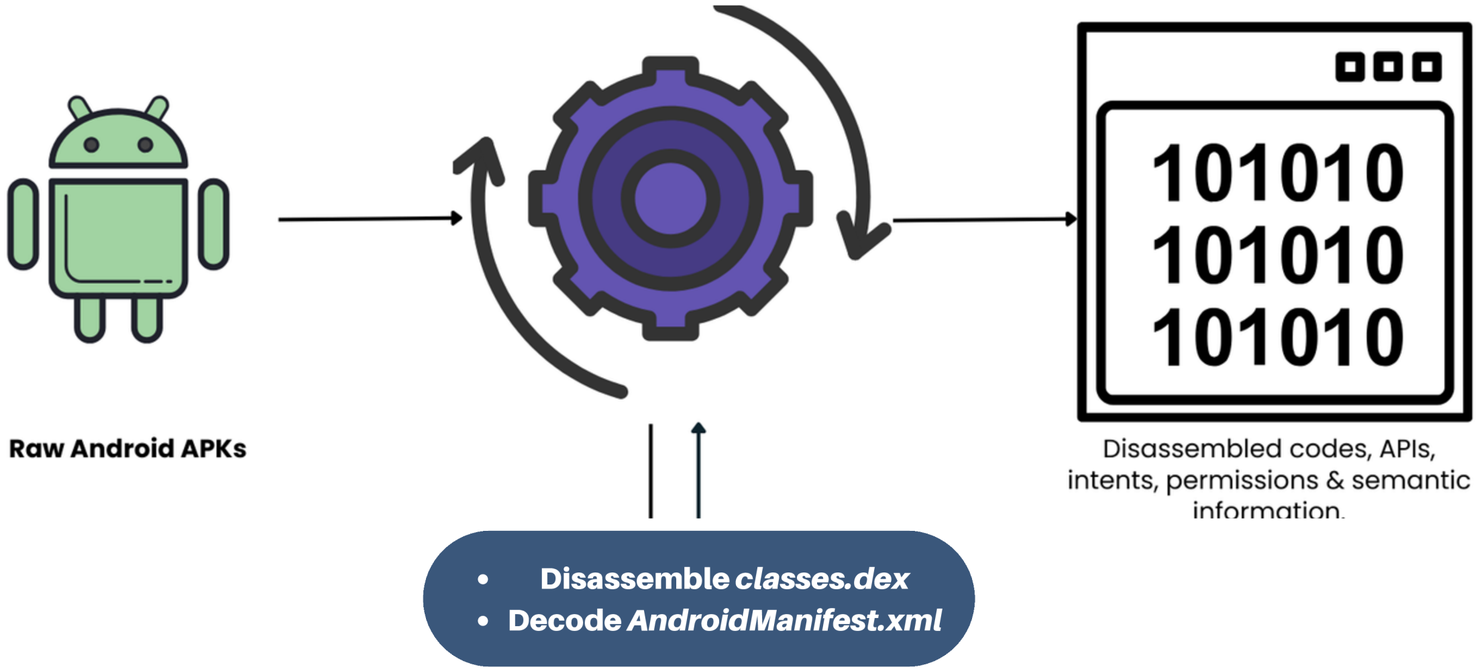
---

**Android APK Components**

- AndroidManifest.xml: Contains metadata like permissions and package details.
- classes.dex: Holds Dalvik bytecode for execution.
- res: Stores resources (images, UI layouts, etc.).
- META_INF: Stores developer signatures for authentication.
- assets: Holds non-compiled resources.

# 2. Reverse Engineering of Android APKs

- Extracting API calls, permissions, intents, and bytecode.
- Features categorized for ML-based models (shallow features) vs DL-based models (deep bytecode analysis).
- Techniques used: Opcode sequence extraction, feature selection, NLP-inspired encoding for bytecode representation.



# 3. Image Conversion

## Byte Stream Formation and Vectorization

- The extracted files are ordered and merged into a single binary stream.
- Each byte is converted into a decimal value (0–255), representing a pixel's grayscale intensity.

## Image Formation

- The one-dimensional array is reshaped into a square 2D matrix to form an image.
- To standardize the input for CNN models, all images are resized to 256×256 pixels using bilinear interpolation (resize() from OpenCV)

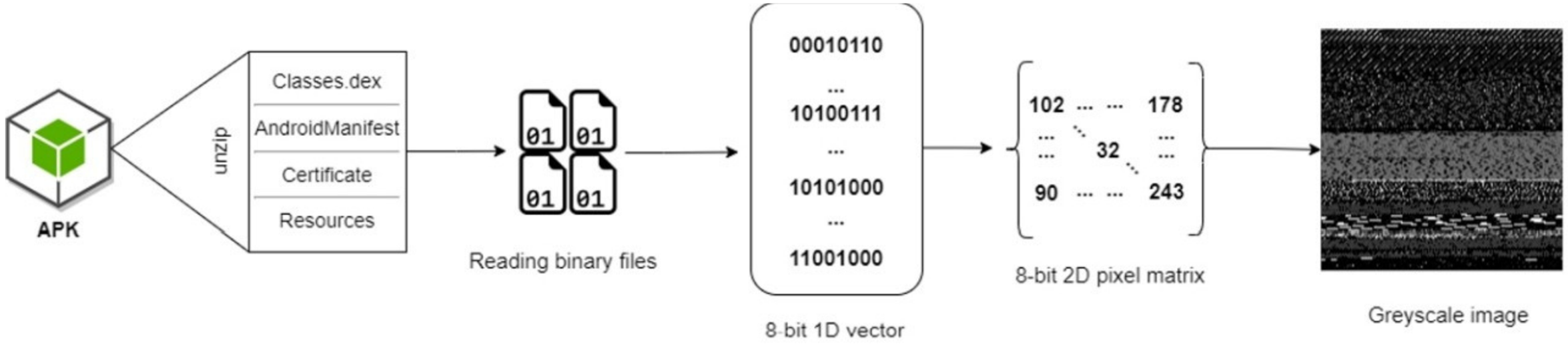
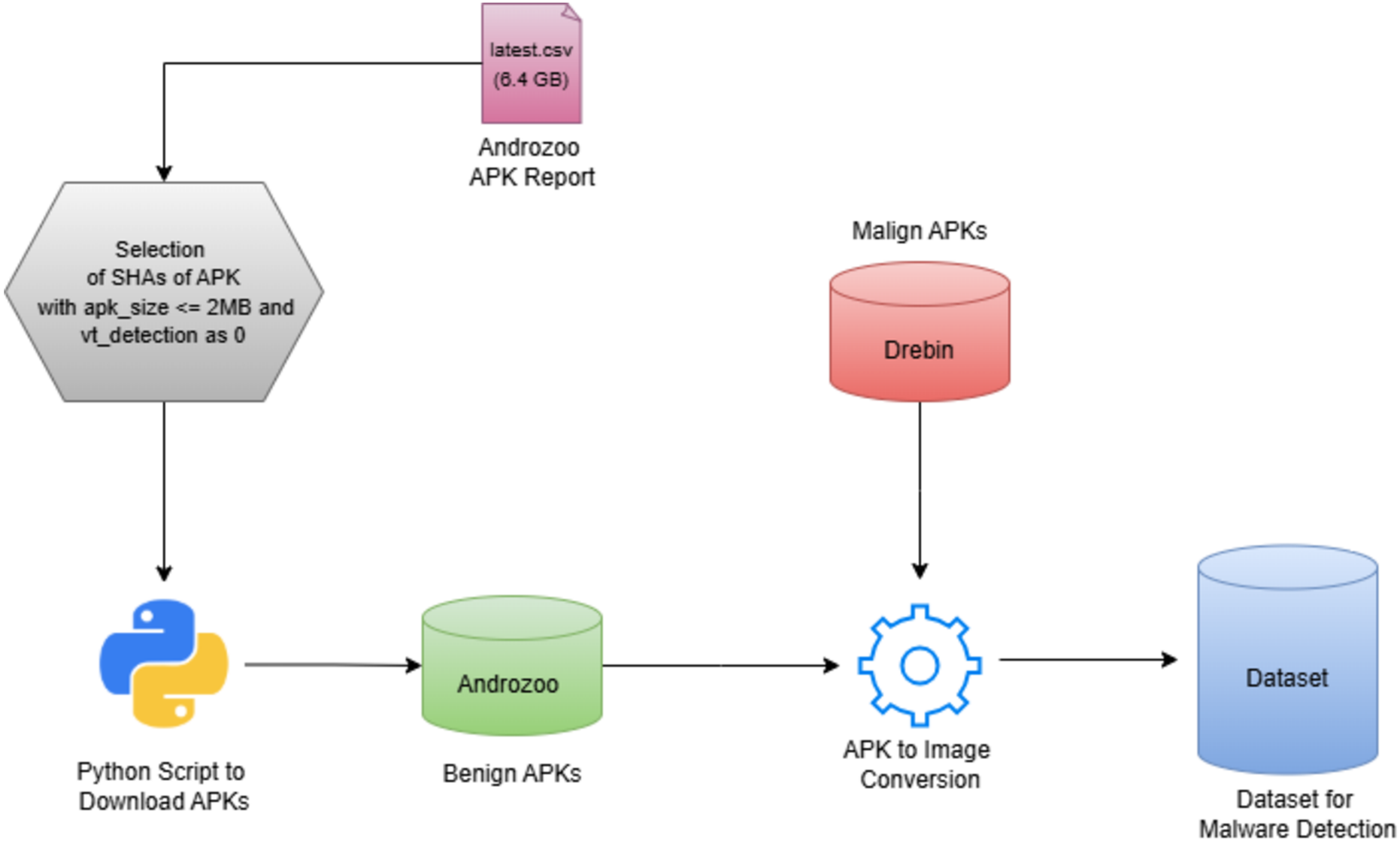



Fig: APK to Image Conversion [10]

# FORMATION OF DATASET



# DEEP NEURAL MODEL ARCHITECTURE

# Convolutional Neural Networks (CNN) :

The CNN is structured in sequential blocks, each aimed at progressively extracting and refining spatial features from the input image.

## CNN Model Architecture:

### Block Design:

**Each of the four blocks consists of:**

- **Two Convolutional Layers having 32 filters with 7×7 kernel activated by ReLU - learn local features (e.g., byte patterns)**
- **Two Batch Normalization Layers having 64 filters with 5×5 kernel - stabilize learning**
- **One MaxPooling Layer (2×2) having 128 convolutional filters with 3×3 kernel - reduces dimensionality, focuses on key patterns**
- **One Spatial Dropout Layer (drop rate: 20%) having 256 separate filters with 3×3 kernel - prevents overfitting**

### Flatten and Dense Layers:

- **A Flatten Layer converts the final feature map into a 1D vector**
- **A Fully Connected Layer (256 neurons) is used for output generation**
- **For binary classification, a Sigmoid activation function is used**

# CNN-LSTM MODEL ARCHICETURE :

To enhance accuracy by modeling sequential dependencies among features:

- **Block Design:**
  - **Block Design is the same as CNN Architecture.**
- **TimeDistributed Layer:**
  - **The CNN is wrapped inside a TimeDistributed layer to allow sequential processing.**
- **LSTM Layer:**
  - **One LSTM layer with 256 units and Tanh activation captures long-term relationships.**
  - **A Dropout Layer (20%) is added to prevent overfitting.**
- **Final Dense Layers:**
  - **For binary detection:**

**A sequence of Dropout Dense(256) L1/L2 Regularization Dropout FinalDense(1).**

## Final Classification Layers:

- **If the model's output is ≥ 0.5, the APK is labeled as malware.**
- **If the model's output is < 0.5, the APK is labeled as benign.**

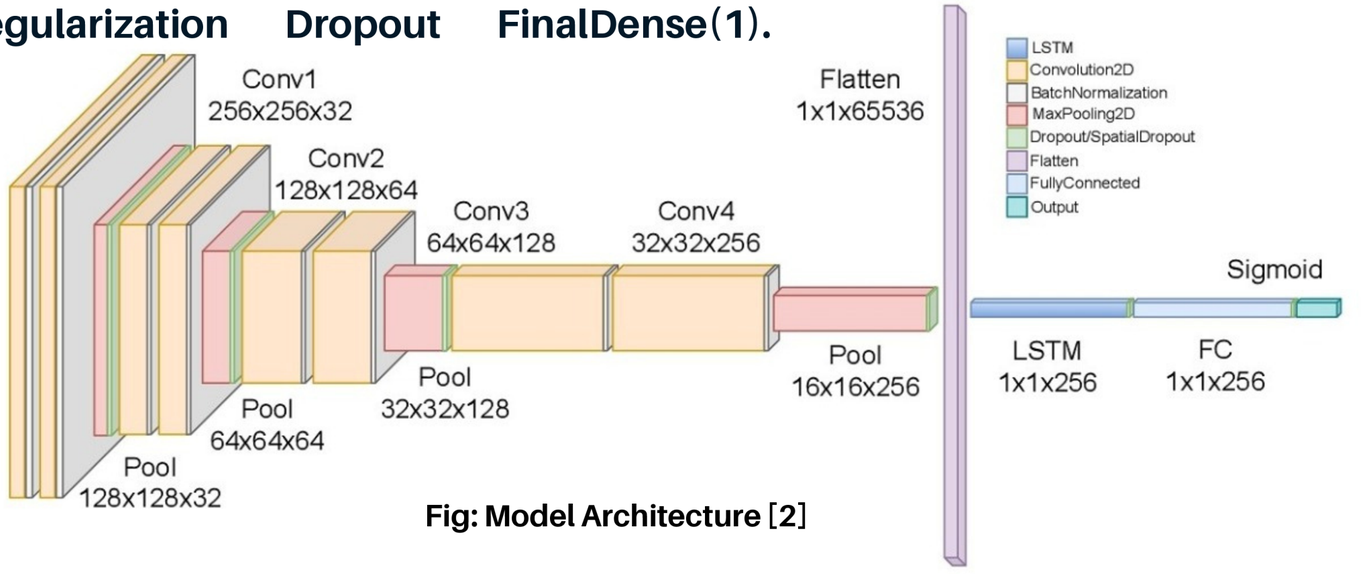
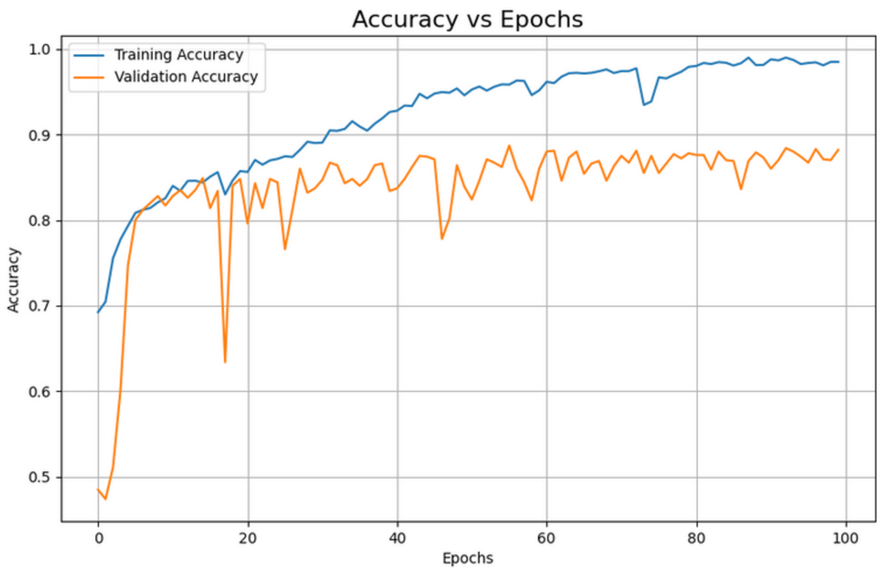



**Fig: Model Architecture [2]**

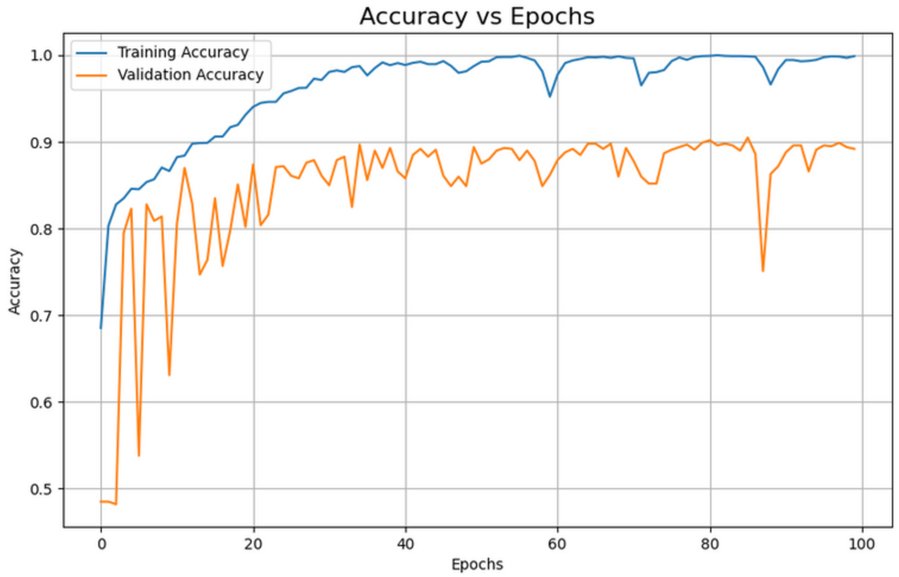
# CODE REVIEW AND DEMONSTRATION

# RESULTS OBTAINED

# Accuracy:




**Accuracy vs Epochs plot of CNN model**



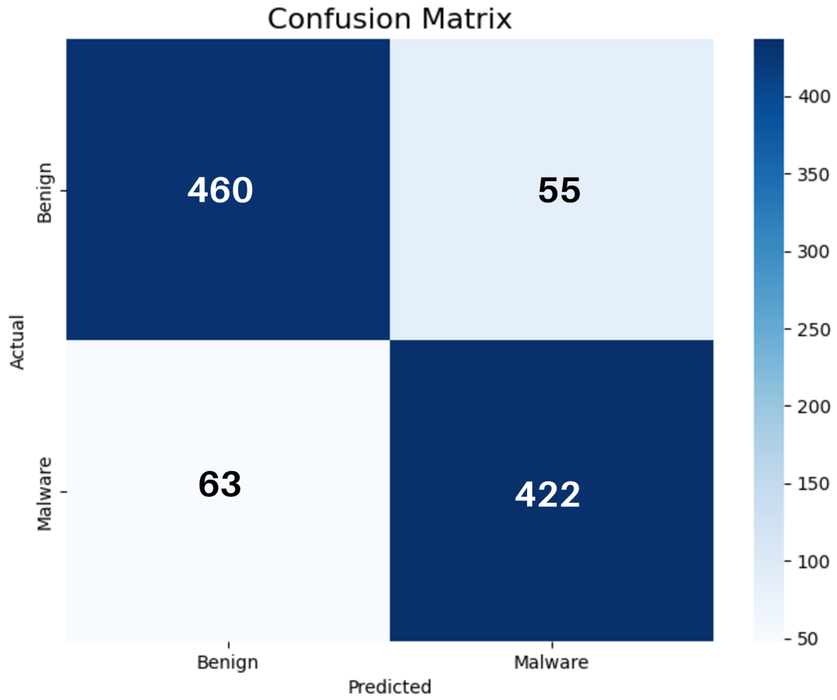

**Accuracy vs Epochs plot of CNN LSTM model**

- Peak training accuracy: 98.97% at Epoch 88.
- Peak validation accuracy: 88.70% at Epoch 56.
- Validation accuracy fluctuated between 82–88.7%, indicating potential overfitting.

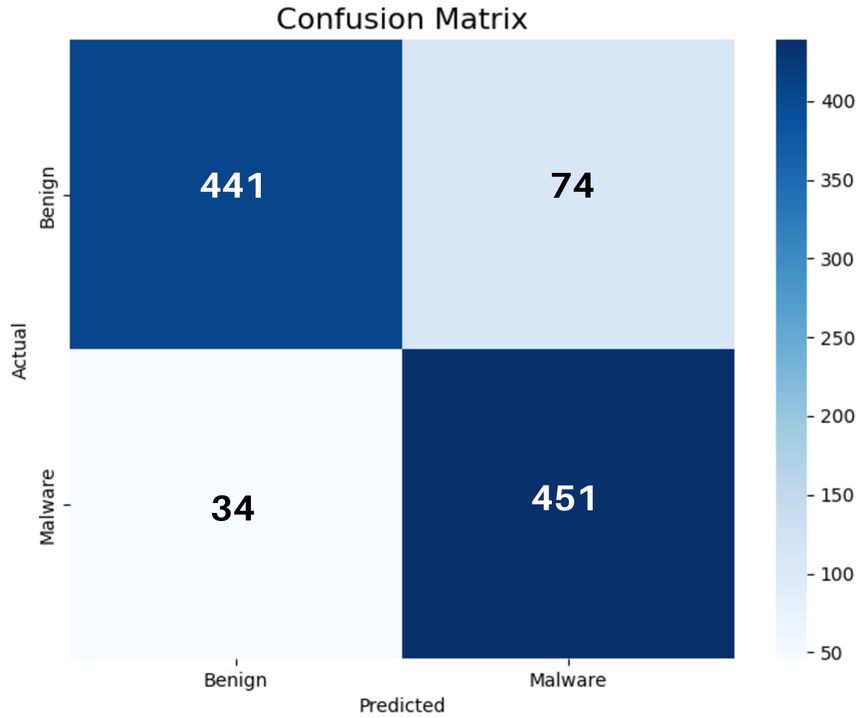
- Peak training accuracy: 100.00% at Epoch 82.
- Peak validation accuracy: 90.50% at Epoch 86.
- Validation accuracy showed some fluctuations but less severe than CNN with no significant overfitting.

- **The CNN model displays signs of overfitting after Epoch 56 by fluctuating validation accuracy and oscillating validation loss (e.g., 2.6092 at Epoch 87).**
- **The CNN-LSTM model demonstrates better resistance to overfitting, maintaining a lower and more stable validation loss (e.g., 0.4299 at Epoch 80) and achieving peak validation accuracy later (Epoch 86), suggesting improved generalization despite its sequential structure.**

# Confusion Matrix:




**CNN**




**CNN-LSTM**

**Lower false positives (55) → Better benign detection**

**Lower false negatives (34) → Better malign detection**

- **In malware detection, minimizing false negatives is often more critical than reducing false positives, giving the CNN-LSTM model an advantage.**

# CNN vs CNN-LSTM

| Aspect | CNN | CNN-LSTM |
|---|---|---|
| Training Accuracy (Max) | 98.97% (Epoch 88) | 100.00% (Epoch 82) |
| Validation Accuracy (Max) | 88.70% (Epoch 56) | 90.50% (Epoch 86) |
| Test Accuracy | 88.20% | 89.20% |
| ROC AUC Score | 0.96 | 0.96 |
| Loss | 0.96 | 0.43 |
| Recall | 87.00% | 93.00% |

- **The CNN-LSTM model outperforms the CNN model with a higher test accuracy (89.20% vs. 88.20%), lower test loss (0.4820 vs. 0.9636), and better recall for malware (0.93 vs. 0.87).**
- **Its ability to reduce false negatives (34 vs. 63) makes it more suitable for malware detection, where missing malware is a critical concern.**

# Future Plan

**Integration with Dynamic Analysis:**

Combine static image-based detection with behavioral features for stronger hybrid detection.

**Multi-Class Malware Classification:**

Extend from binary classification to identifying specific malware families for improved threat analysis.

**Real-Time & Lightweight Deployment:**

Optimize CNN architecture for faster inference and lower memory usage, enabling deployment on mobile/edge devices.

# Conclusion

- CNN-LSTM significantly reduced training time while preserving high detection performance.
- Eliminated the need for domain-specific manual feature extraction—purely image-driven approach.
- Achieved approx. 90% accuracy using CNN-LSTM on a balanced dataset of 5,000 apps.
- Converted APK components to grayscale images to extract spatial features using CNNs.

# References


[1] Nataraj, Lakshmanan, et al. ”Malware images: visualization and automatic classification proceedings of the 8th international symposium on visualization for cyber security. 2011.

[2] Soman, Akarsh, et al. A Detailed Investigation and Analysis of Deep Learning Architectures and Visualization Techniques for Malware Family Identification. 06 2019, pp. 241–286,https://doi.org/10.1007/978-3-030-16837-7 12.s.

[3] Almomani, Iman, Aala Alkhayer, and Walid El-Shafai. ”E2E-RDS: Efficient End-to-End ransomware detection system based on Static-Based ML and Vision-Based DL approaches.”Sensors 23.9 (2023): 4467.

[4] Kumar, Rajeev, Neeraj Kumar, and Ki-Hyun Jung. ”Color image steganography scheme using gray invariant in AMBTC compression domain.” Multidimensional Systems and Signal Processing 31.3 (2020): 1145-1162.

[5] Vasan, Danish, et al. ”IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture.”Computer Networks 171 (2020): 107138.

[6] Hsien-De Huang, TonTon, and Hung-Yu Kao. ”R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections.” 2018 IEEE international conference on big data (big data). IEEE, 2018.

[7] Vu, Long & Jung, Souhwan. (2021). AdMat: A CNN-on-Matrix Approach to Android Malware Detection and Classification. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3063748.

[8] Yadav, Pooja, et al. ”EfficientNet convolutional neural networks-based Android malware detection.” Computers & Security 115 (2022): 102622.

[9] Qiu, Junyang, et al. ”A survey of android malware detection with deep neural models.” ACM Computing Surveys (CSUR) 53.6 (2020): 1-36.

[10] Aldini, Alessandro, and Tommaso Petrelli. ”Image-based detection and classification of Android malware through CNN models.” Proceedings of the 19th International Conference on availability, Reliability and Security. 2024.

[11] Allix, Kevin, et al. ”Androzoo: Collecting millions of android apps for the research com-munity.” Proceedings of the 13th international conference on mining software repositories 2016.

[12]Kabakus, Abdullah Talha. ”DroidMalwareDetector: A novel Android malware detection framework based on convolutional neural network.” Expert Systems with Applications 206(2022): 117833.

[13] Sherif, Ahmed. “Android - Statistics &Amp; Facts.” Statista, 27 Sept. 2024,www.statista.com/topics/876/android.

# THANK YOU!