AndroBlight : Android Malware Detection using CNN

Presented by

Group ID: 47

Branch: ECSc

Aditya Sarkar (2230006)

Akash Kumar (2230008)

Ashmit Dutta (2230156)

Asish Kumar (2230240)

Supervisor: Prof. Sandeep Kumar Dash

School of Electronics Engineering

KIIT Deemed to be University, Bhubaneswar

INTRODUCTION

Android's widespread adoption and open-source nature have made it a prime target for increasingly sophisticated malware attacks. Traditional detection methods struggle to keep pace with these evolving threats, creating a critical need for more robust and adaptive solutions. This presentation introduces "Androblight," a project focused on the correct identification of Android malware using advanced deep learning models. We will explore a multi-step framework that leverages raw Android APK data and reverse engineering techniques, integrated with app representation learning, to train a customized Convolutional Neural Network (CNN) for precise malware classification. Our goal is to enhance malware detection precision and adaptability, providing an efficient and scalable defense mechanism for the Android ecosystem.

LITERATURE SURVERY

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 Hybrid Models:

- Many CNN hybrid models have given much better result as compared to using standalone CNN.

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: DroidMalware Detector, 90% accuracy (14,000-app dataset) [12].

OBJECTIVE

Problem Addressed:

- Correct identification of Android malware using deep learning models.

Objective:

- Develop a static analysis system for Android malware detection using deep learning models.
- Leverage CNNs and CNN-LSTM models to automatically extract features from APK files.
- Achieve high accuracy, precision, recall, and F1-scores on benchmark datasets.
- Implement it as a full-stack application using Python for the backend and React.js for the frontend.

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.

DATA FLOW OF ML MODEL

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. Collection and pre-processing of apks

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.

Dataset Used :

- AndroZoo (Benign APKs):
  - Large dataset (24M+ APKs).
  - Rich metadata (permissions, API calls, etc.).
  - Enables research reproducibility.
  - Covers diverse app categories.
- Drebin Dataset (Malware APKs):
  - Contains 5,560 malware samples across 179 families.
  - Useful for benchmarking malware detection methods.

Challenges Faced:

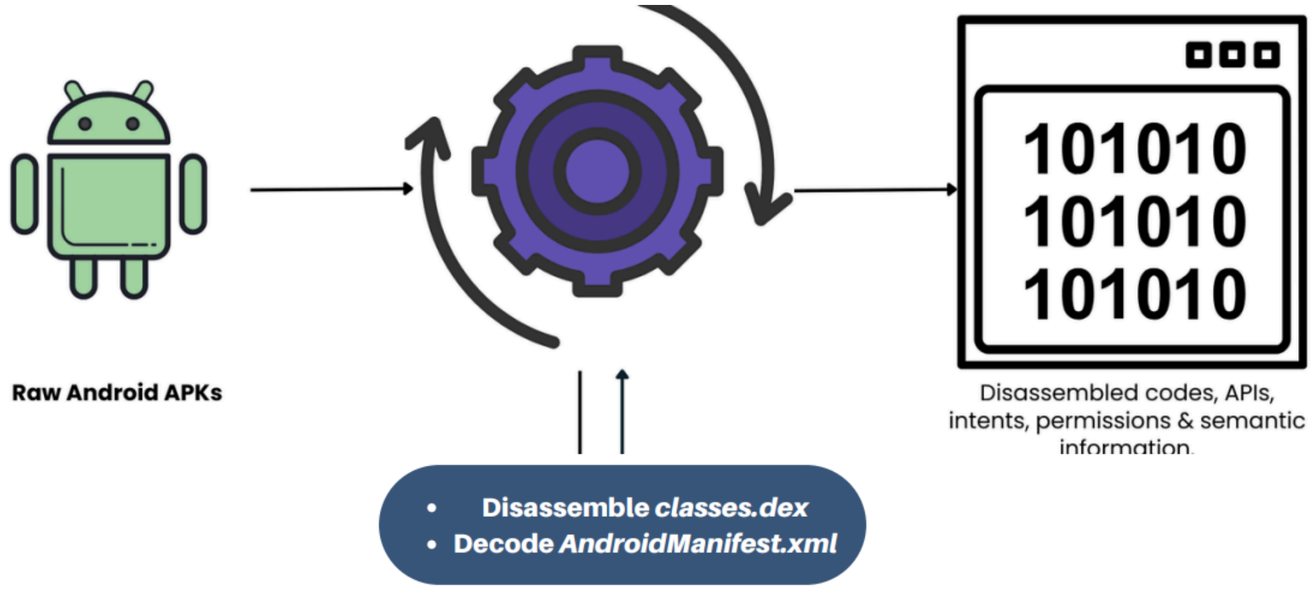
- Access Restrictions: Some datasets require special permissions for use.
- Data Volume & Complexity: Handling millions of APKs requires high computational resources.
- API Limits: Rate-limited APIs hinder large-scale data collection.
- Aging Dataset: Some malware datasets may not reflect recent threats.

IMPLEMENTATION DETAIL

2. Reverse Engineering of Android APKs

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.
- Reverse engineering tools include TaintDroid, Andrubis, MARVIN, Mobile-Sandbox.



IMPLEMENTATION DETAIL

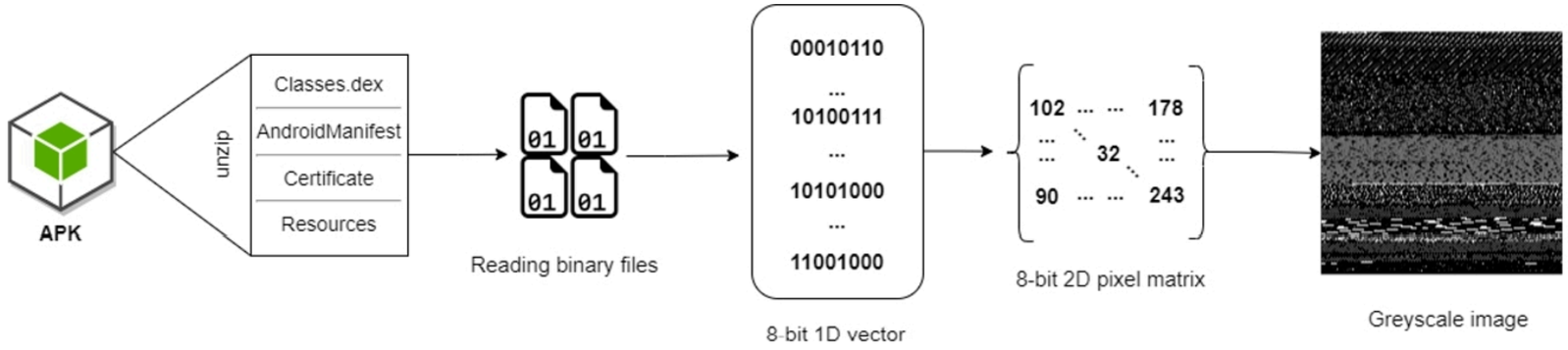
3. Neural Model Adaptation & Evaluation

Neural Model Adaptation & Evaluation:

- DL models learn high-level feature representations for malware detection.
- CNN+LSTM hybrid model used for sequential learning of APK behavior.

Evaluation Metrics:

- True Positive (TP): Correctly identified benign apps.
- True Negative (TN): Correctly identified malware apps.
- False Positive (FP): Benign apps misclassified as malware.
- False Negative (FN): Malware apps misclassified as benign.
- Accuracy, Precision, Recall, and F1-score calculated to measure model performance.



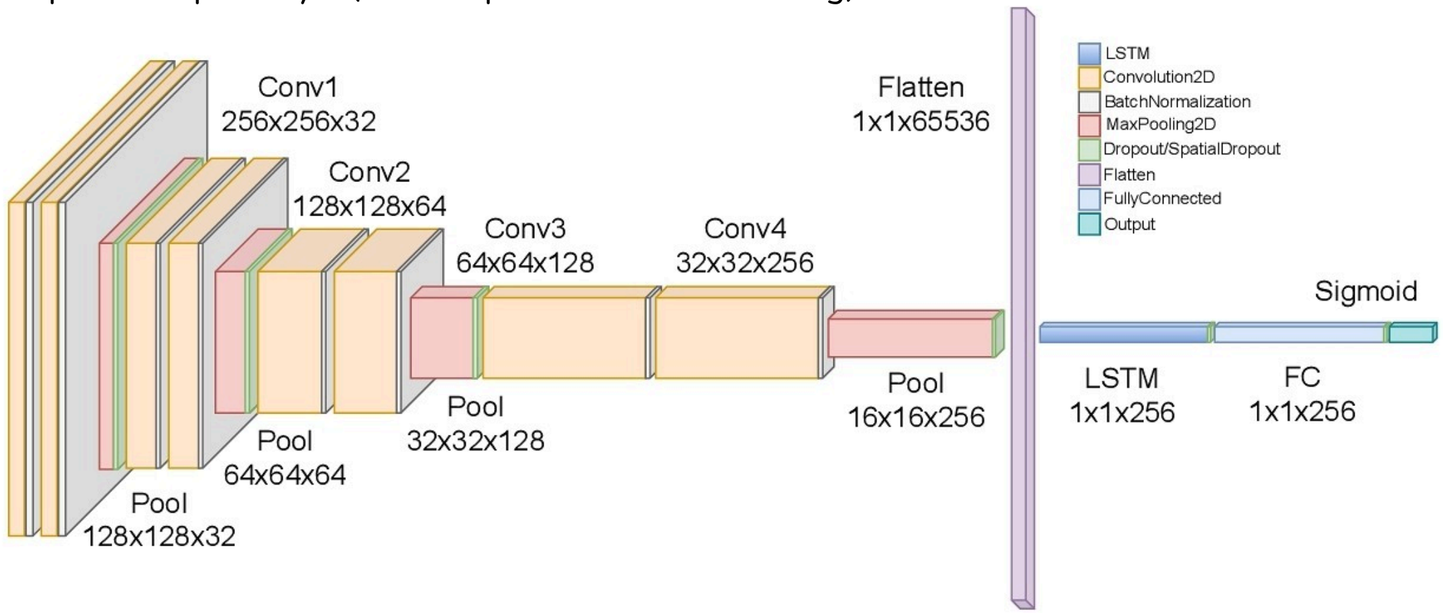
ALGORITHMS USED

Convolutional Neural Networks (CNNs):

- Specialized in grid-like data structures
- Ideal for computer vision and malware detection.
- Extracts hierarchical features from API-call sequences.
- Uses ReLU activation for non-linearity and feature learning.

Base CNN Architecture:

- Four convolutional blocks, each containing:
  - Two Convolutional layers (ReLU activation).
  - Two Batch Normalization layers.
  - Max Pooling layer (2×2 filter for downsampling).
  - Spatial Dropout layer (20% dropout to avoid overfitting).



ALGORITHMS USED

CNN-LSTM Extension for Sequence Learning:

- CNN extracts spatial patterns from images/APK features.
- LSTM processes sequences to learn long-range dependencies.
- TimeDistributed layer ensures CNN outputs are sequentially fed to LSTM
- LSTM layer consists of 256 neurons with tanh activation.
- Dropout (20%) used for regularization.

Final Classification Layers:

- Fully connected (dense) layers for final classification.
- Dropout layer (50%) for overfitting prevention.
- Dense layer (256 neurons) with L1/L2 regularization.
- Final output layer with Sigmoid activation for binary classification..

WEB-APP IMPLEMENTATION

Frontend (React + Tailwind):

- Axios integration: Secure POST to /predict with FormData, shows prediction, confidence, and (for malign) and to find malware family label for demo and UX continuity.
- Metrics UI: Cards for Benign / Malign counts, and model comparison cards (tables) for Precision, Recall, F1, Accuracy for CNN vs CNN-LSTM.
- Loss/Accuracy charts: Line/area overlays using ApexCharts with custom y-axis ranges (0–1) and toggleable series.
- Confusion matrix views: Side-by-side tiles for CNN and CNN-LSTM.

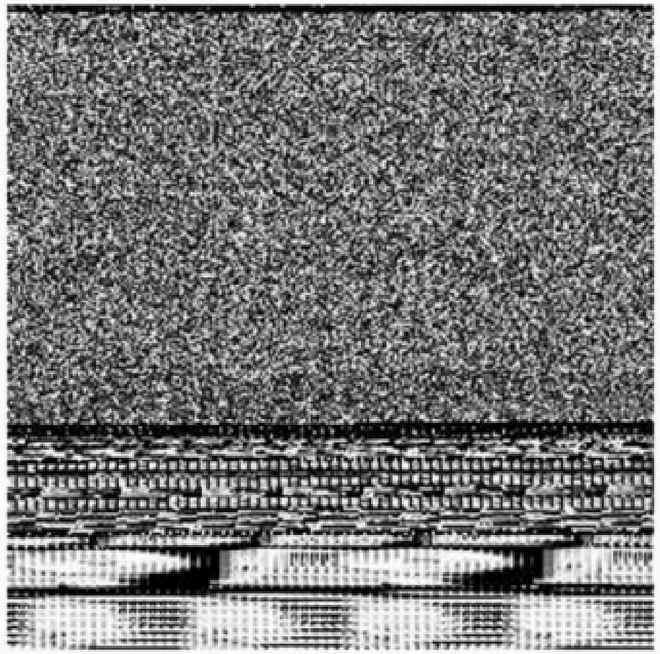
Backend (Flask API):

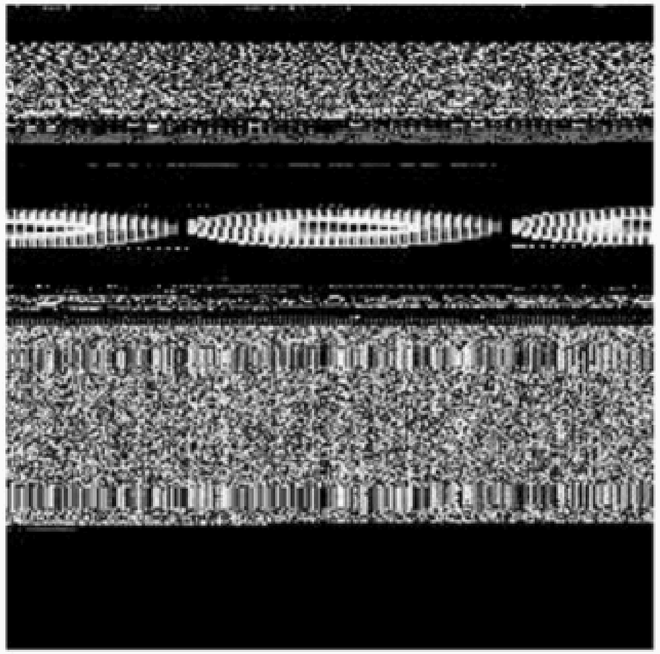
- Endpoint: POST /predict accepts .apk via multipart form data.
- APK processing: Extracts selected files (classes.dex, AndroidManifest.xml, META-INF/CERT.RSA, resources.arsc), concatenates bytes → padded square → 256×256 grayscale tensor.
- Model inference: Loads cnn-lstm_detection_model.h5; binary output → Benign/Malign + confidence.
- CORS: Scoped to the frontend origin.
- Cleanup: Removes uploaded APK and temp extract dir after each request.
- Health check: GET /health.

PROTOTYPE DEMONSTRATION

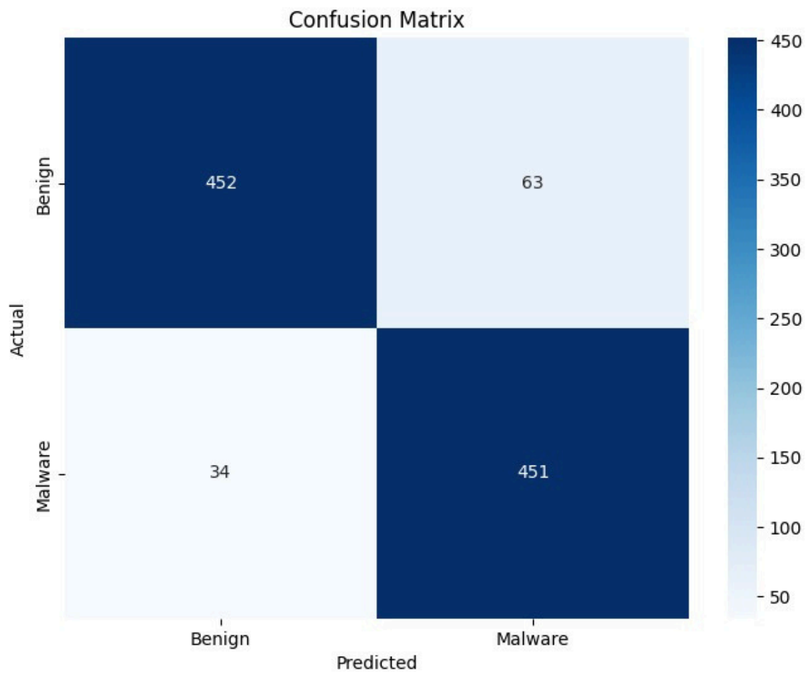
RESULT

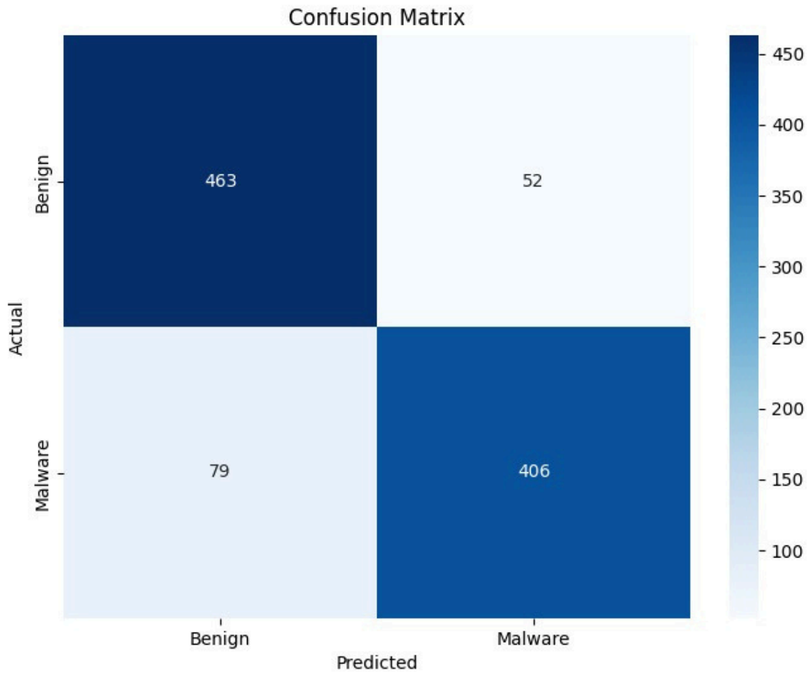
RESULT OBTAINED





BENIGN (LEFT) & MALIGN(RIGHT)

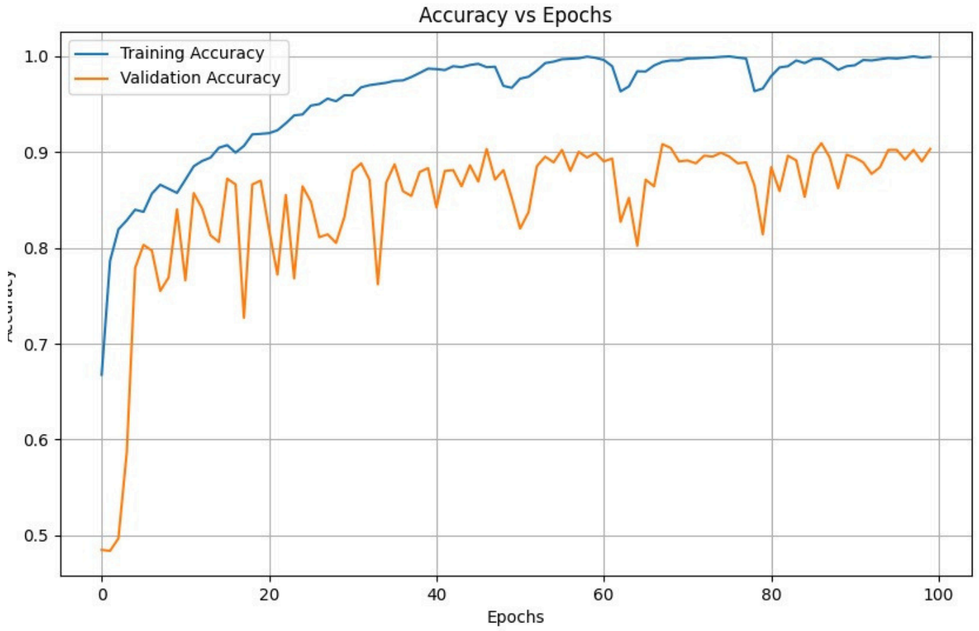




CNN

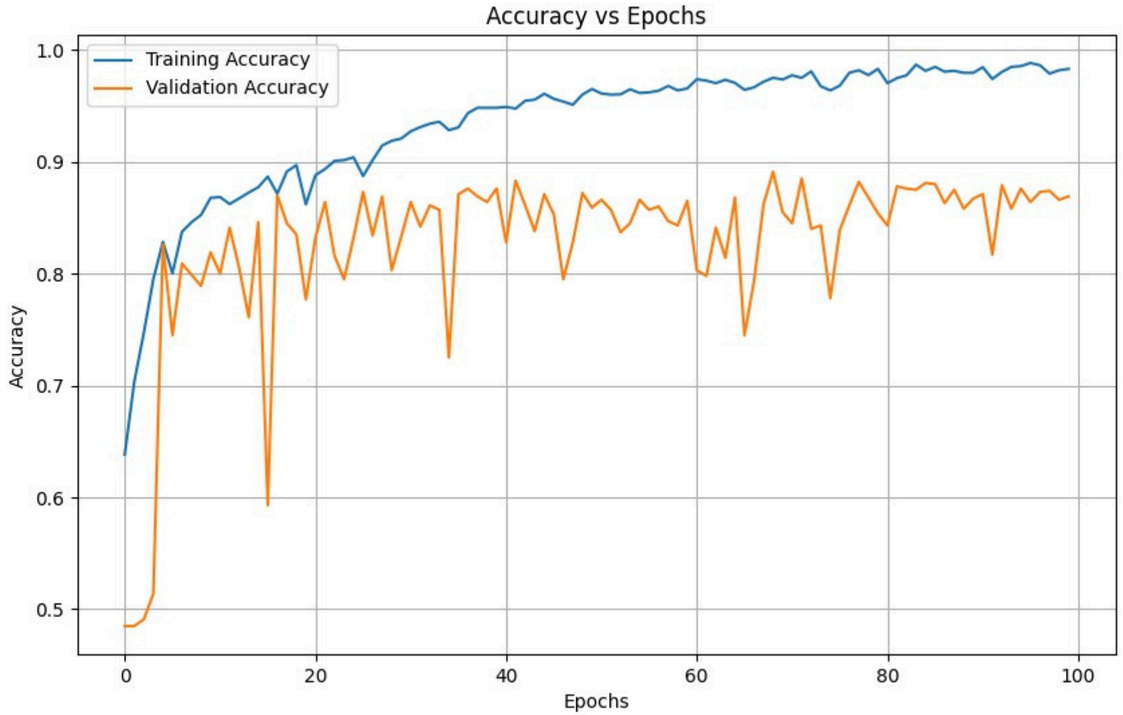
CNN-LSTM

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

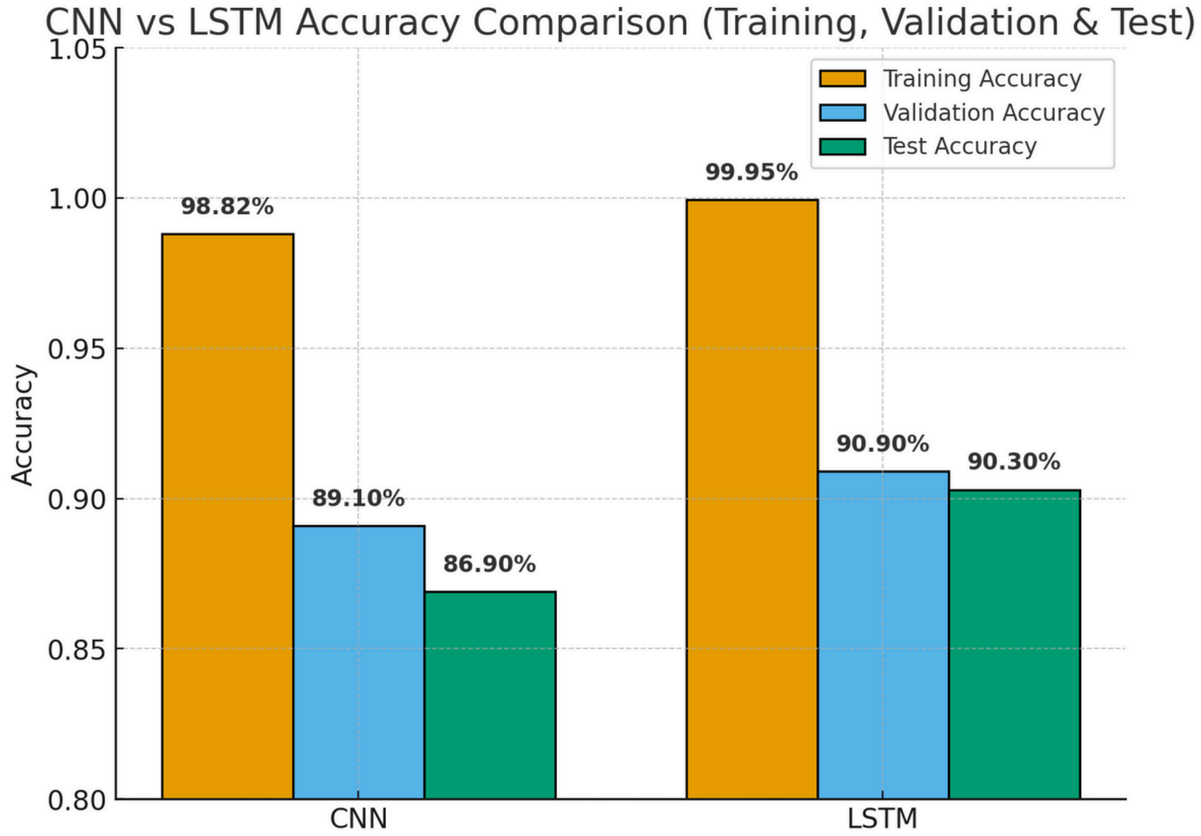

Accuracy vs Epochs
Training Accuracy
Validation Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
Epochs

CNN-LSTM

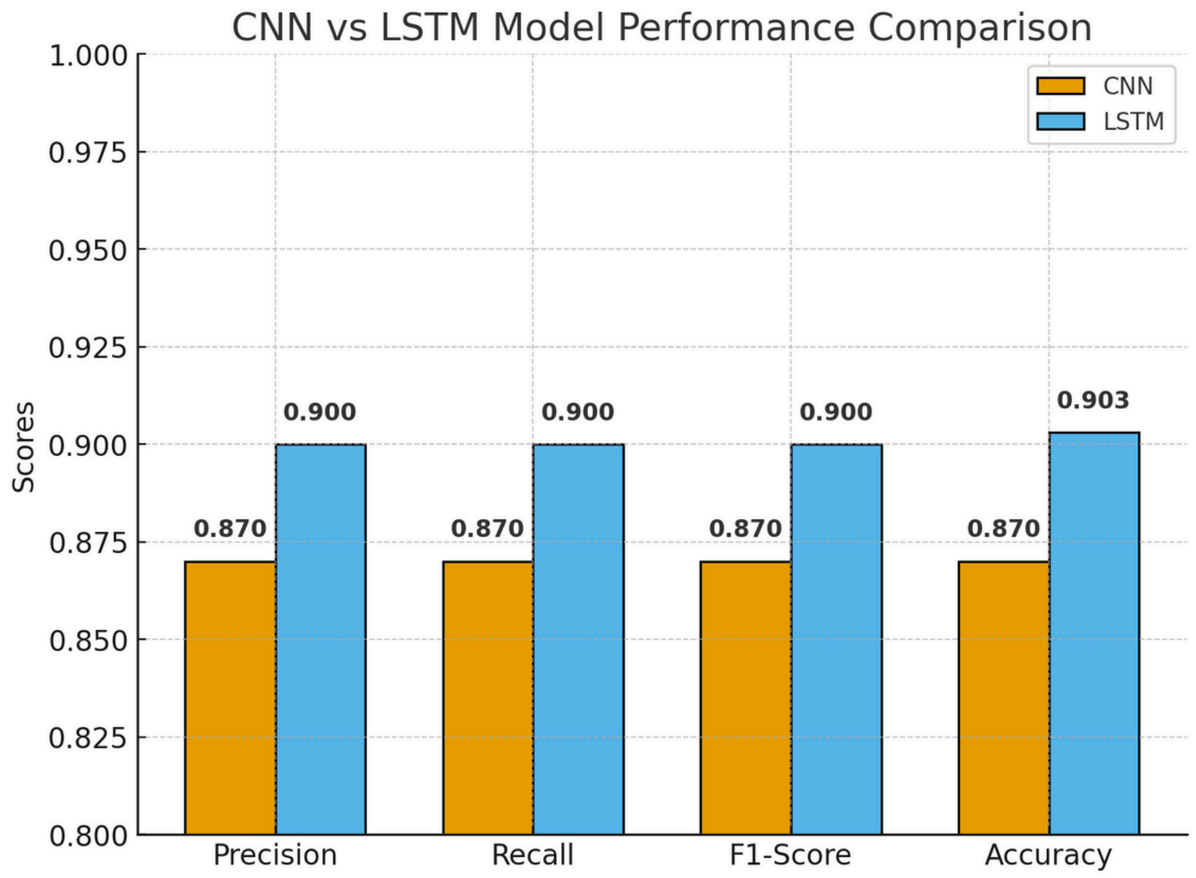
CNN


Accuracy vs Epochs
Training Accuracy
Validation Accuracy
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
Epochs

CNN VS CNN-LSTM



CNN VS CNN-LSTM



WORK TO BE CARRIED OUT (8^{TH} SEMESTER)

- Real Time & light weight App Deployment: Optimize CNN architecture for faster inference and lower memory usage, enabling deployment on mobile devices with easy to use UI.
- Extend the Binary classification to identify specific malware family for improved threat analysis.
- Work on many more malware classes so to further enhance the malware identification.
- Deploying web-application and android application LIVE for the use of general people.
- Writing IEEE papers for the respective project.

WORK DONE BY MEMBERS

BY ADITYA SARKAR

Key Contributions:

- Designed and implemented the metrics visualization dashboard for the project “Malware Detection using CNN”.
- Developed interactive charts to display key model performance metrics such as accuracy, loss, and confusion matrix.
- Ensured a user-friendly interface for better understanding of model evaluation and results

Tools & Technologies Used:

- React.js, Chart.js / Apexcharts
- HTML, CSS, JavaScript, Tailwind

Outcome:

- Delivered a responsive and visually informative dashboard that clearly represents the model’s performance insights.

BY AKASH KUMAR

Key Contributions:

- Designed and implemented both CNN and hybrid CNN-LSTM models for malware detection.
- Conducted model training, evaluation, and optimization to enhance accuracy and generalization.
- Tuned hyperparameters, applied regularization, and improved dataset preprocessing for better model performance.
- Analyzed and compared model metrics such as accuracy, loss, and confusion matrix.

Tools & Technologies Used:

- Python, TensorFlow / Keras, NumPy, OpenCV, Jupyter Notebook

Outcome:

- Achieved significant improvement in model accuracy and stability through optimized CNN and CNN-LSTM architectures.

BY ASHMIT DUTTA

Key Contributions:

- Handled data preparation and preprocessing for the malware detection dataset.
- Extracted and organized APK files, performed data cleaning, and converted APKs into grayscale image representations in Google Colab.
- Supported backend integration by ensuring smooth data flow between preprocessing scripts and the model API.
- Ensured data consistency and quality to enhance the accuracy and reliability of model training.

Tools & Technologies Used:

- Python, Google Colab, NumPy, OpenCV, Flask

Outcome:

- Provided a clean and structured dataset that significantly improved model performance and facilitated efficient backend processing.

BY ASISH KUMAR

Key Contributions:

- Developed the backend using Flask to handle APK upload, processing, and model inference.
- Integrated the frontend (React.js) with the backend to present the working malware detection model seamlessly.
- Enabled real-time APK analysis, displaying prediction results and model responses dynamically.
- Ensured smooth communication between the ML model and user interface for efficient performance.

Tools & Technologies Used:

- Flask, React.js, REST API, Python, HTML, CSS, JavaScript

Outcome:

- Delivered a fully functional end-to-end system that allows real-time APK malware detection through an interactive web interface.

REFERENCES

1. [1] Nataraj, Lakshmanan, et al. "Malware images: visualization and automatic classification proceedings of the 8th international symposium on visualization for cyber security. 2011.
2. [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. [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. [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. [5] Vasan, Danish, et al. "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture."Computer Networks 171 (2020): 107138.
6. [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.

REFERENCES

1. [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.
2. [8] Yadav, Pooja, et al. "Efficient Net convolutional neural networks-based Android malware detection." Computers & Security 115 (2022): 102622.
3. [9] Qiu, Junyang, et al. "A survey of android malware detection with deep neural models." ACM Computing Surveys (CSUR) 53.6 (2020): 1-36.
4. [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.
5. [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.
6. [12] Kabakus, Abdullah Talha. "DroidMalwareDetector: A novel Android malware detection framework based on convolutional neural network." Expert Systems with Applications 206(2022): 117833.
7. [13] Sherif, Ahmed. "Android - Statistics &Amp; Facts." Statista, 27 Sept. 2024,www.statista.com/topics/876/android.

CONCLUSION

- Deep Learning is transforming malware detection by providing advanced data representations and abstractions.
- Research trends show a shift towards improved neural architectures capable of identifying complex malware patterns.
- The use of CNNs and other deep learning architectures enhances classification accuracy and adaptability.
- These advancements will serve as a foundation for solving the growing challenges in Android malware detection.
- Future developments in DL-based malware detection will continue to enhance mobile security and threat mitigation.

THANK YOU