Contents lists available at IdNNS
Indonesian Journal of Cybersecurity and Emerging Risks
Volume 1 ; Issue 1
journal homepage: https://journal.idnns.org/index.php/ijcyber

Accuracy Comparison of Hybrid CNN-BiLSTM and Machine Learning Models in Android Malware Detection

Ahmad Muzakir Aulia Musanit[a,i]

[a]*Information Master Study Program, Nusa Putra University, Sukabumi, Indonesia*

ARTICLE INFO

Keywords:
Android
malware
detection
Hybrid CNN-BiLSTM
machine learning
permissions
API
cybersecurity

ABSTRACT

Malware is malicious software designed to damage computer and mobile devices by accessing, stealing personal information, or taking control of devices through networks. This threat is especially severe on Android platforms, as it can disrupt device performance and compromise user data. Therefore, effective and reliable malware detection methods are required. This research aims to develop and evaluate a malware detection method for Android devices using a Hybrid Convolutional Neural Network (CNN+BiLSTM) approach. The dataset used in this study was sourced from TUANDROMD (Tezpur University Android Malware Dataset). The results indicate that the Hybrid CNN-BiLSTM method achieves a higher accuracy than other machine learning methods. These findings suggest that this hybrid approach is one of the solutions that can be used to improve the security of Android devices against malware attacks compared to CNN and other machine learning methods in this study.

1. Introduction

The Android operating system, originally developed by Android Inc. in 2003 and later acquired by Google in 2005, has become the dominant platform for mobile devices worldwide. Since its official release in 2008, Android has continuously evolved to meet the demands of modern users, incorporating significant innovations such as Material Design and enhanced security frameworks. By 2023, Android accounted for approximately 48.22% of the global population of mobile users, with a particularly high market share exceeding 87% in Indonesia. This ubiquity has made Android a prime target for malicious actors, who exploit its open-source architecture and extensive application ecosystem to distribute malware through various attack vectors, including malicious applications and system vulnerabilities.

Malware, or malicious software, encompasses a wide range of harmful programs designed to infiltrate, disrupt, or gain unauthorized access to mobile devices. Variants such as trojans, ransomware, spyware, and adware pose significant threats to user privacy, financial security, and overall system performance. As malware authors increasingly employ obfuscation techniques and adaptive behaviors to evade detection, traditional signature-based and heuristic detection methods have proven insufficient in addressing these evolving threats. Consequently, the research community has turned to machine learning (ML) and deep learning (DL) approaches, which leverage statistical patterns and complex feature representations to improve detection accuracy and adaptability.

Existing ML and DL-based malware detection methods, while effective, exhibit limitations in modeling sequential and temporal dependencies inherent in malicious behaviors. Most conventional techniques rely heavily on static analysis of permissions and API calls, neglecting the dynamic patterns that manifest over time. To address this gap, hybrid deep learning architectures have been proposed, combining spatial feature extraction with temporal sequence modeling. Among these, the integration of Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks offers a compelling framework. CNNs excel at capturing localized spatial patterns, while BiLSTM networks are well-suited for learning bidirectional temporal dependencies.

This study proposes a Hybrid CNN-BiLSTM model for Android malware detection, designed to leverage the complementary strengths of both architectures. The hybrid model extracts spatial features from Android application attributes using CNN layers and models

[i]Corresponding author
Ahmad.muzakir@nusaputra.ac.id (A.M.A. Musanit)
ORCID(s):

sequential dependencies through BiLSTM layers, enabling robust classification of benign and malicious applications. The proposed framework is evaluated on the publicly available TUANDROMD dataset from Tezpur University, which contains 4,465 samples characterized by permission- and API-based features. To address the inherent class imbalance in the dataset (malware-to-benign ratio of approximately 4:1), the Synthetic Minority Over-sampling Technique (SMOTE) is applied, ensuring balanced training data and mitigating bias toward the majority class.

The primary contributions of this research are as follows:

- Development of a hybrid deep learning model combining CNN and BiLSTM for accurate Android malware classification;
- Comprehensive evaluation of the proposed model using accuracy, precision, recall, and F1-score under 5-fold stratified cross-validation;
- Comparative analysis of the hybrid model against standalone CNN and conventional ML algorithms, including Support Vector Machine (SVM) and Logistic Regression, to validate performance improvements.

Through this approach, the study aims to advance Android malware detection by providing a robust, scalable framework capable of addressing evolving threats in real-world mobile ecosystems.

2. Research Methodology

2.1. Dataset Overview

The dataset used in this study is the TUANDROMD dataset, developed by Tezpur University. It is a benchmark dataset widely utilized for research in Android malware detection leveraging machine learning and deep learning methodologies. The dataset comprises 4,465 Android applications with 241 extracted features derived primarily from application permissions and API calls, categorized as either benign or malware in the target label column. Each feature is represented numerically, predominantly in binary form (0/1) or float64, indicating the absence or presence of specific attributes.

A distinctive feature of TUANDROMD is its class imbalance, with approximately 79.84% of samples labeled as malware and 20.13% labeled as benign. This imbalance poses challenges for model generalization and can bias learning toward the majority class. To address this issue, we employed a hybrid resampling strategy that combined Random Undersampling of the majority class and Synthetic Minority Oversampling Technique (SMOTE) to enhance the representation of the minority class during training, as recommended by recent studies (Thakur et al., 2024). This approach was implemented using the imbalanced-learn library in Python.

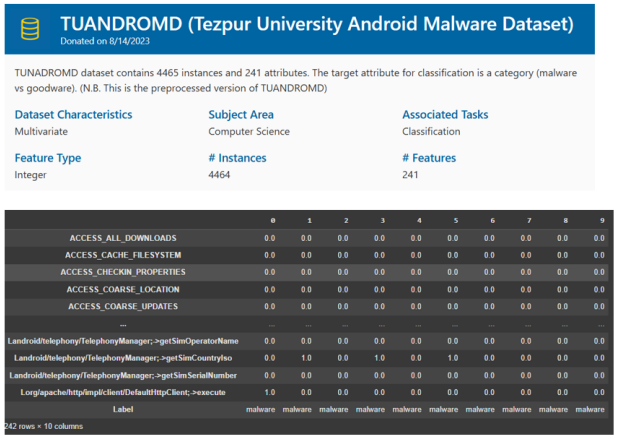

Figure 1: Dataset TUANDROMD

2.2. Data Pre-processing

The data preprocessing phase encompassed several sequential steps to ensure the dataset was clean, standardized, and properly formatted for input into the deep learning architecture. The process began with dropping missing values (N/A), where an initial examination of each attribute identified unique values and potential missing entries (NaN). To maintain data quality and prevent bias during training, rows containing missing values were removed using the `dropna()` method, as the proportion of missing data was negligible and did not significantly affect the dataset's distribution. Following this step, categorical encoding was applied to the target attribute (Label) using scikit-learn's Label Encoding, converting string classes such as "benign" and "malicious" into numerical values (0 and 1). This transformation preserved the categorical nature of the data without expanding feature dimensions, making it suitable for binary classification.

Next, a string-to-integer conversion was conducted for all feature columns to ensure homogeneity in data types, enabling smooth processing by machine learning algorithms. Each attribute was transformed into integer form using the `astype(int)` method, followed by verification of value distributions and unique class counts using `np.unique()`, which was crucial for determining the number of output units in the classification layer. Subsequently, feature standardization was performed using Z-score normalization through the `StandardScaler` object from scikit-learn. This process adjusted each feature to have zero mean and unit variance, ensuring comparability across features and preventing bias toward attributes with larger numerical ranges.

After standardization, the dataset underwent data transformation to meet the input requirements of the Convolutional Neural Network (CNN) component in the hybrid model. Specifically, the two-dimensional standardized feature matrix ($n\_samples$, $n\_features$) was reshaped into a three-dimensional tensor ($n\_samples$,

n_features, 1) using NumPy's `np.expand_dims()` function. This transformation added a channel dimension required by Conv1D layers to correctly process sequential data.

Finally, the preprocessed data was split into training and testing subsets using the `train_test_split()` function from scikit-learn, allocating 80% of the samples for training and 20% for testing, with `random_state=0` to ensure reproducibility. This division provided a dedicated hold-out set for final evaluation, enabling assessment of the model's generalization capability on previously unseen data. The resulting training and testing shapes were verified to confirm compatibility with the subsequent hybrid CNN-BiLSTM training pipeline.

2.3. Model Architectures

The proposed Hybrid CNN-BiLSTM architecture integrates convolutional and recurrent layers to effectively capture both spatial and temporal patterns in Android malware data. The Conv1D layers serve as the initial feature extractors, utilizing two convolutional layers with a kernel size of three to identify local spatial patterns, such as byte sequences or opcodes, that are indicative of malicious behavior. These layers are followed by MaxPooling1D operations, which reduce the dimensionality of the convolutional outputs while retaining critical features, thereby lowering computational complexity and mitigating overfitting. To further enhance generalization, Dropout layers are applied after both convolutional and dense layers, randomly deactivating neurons during training to prevent overreliance on specific features. The core temporal modeling is handled by a Bidirectional LSTM layer, which processes the feature sequences in both forward and backward directions, allowing the model to learn long-range dependencies and contextual relationships from the CNN-extracted features. Subsequently, the output is passed through a Flatten layer to convert the three-dimensional BiLSTM output into a one-dimensional vector suitable for dense processing. Finally, the architecture employs dense layers with ReLU activation, followed by an output layer using sigmoid or softmax activation depending on the classification type (binary or multi-class). The model is compiled with either binary cross-entropy or categorical cross-entropy as the loss function, aligning with the classification objectives. This combination of CNN and BiLSTM layers provides a robust framework for accurately detecting Android malware by leveraging both local feature extraction and sequential dependency modeling.

In addition to this hybrid model, classic machine learning models are also used for comparison, namely Support Vector Machine (SVM) and Logistic Regression. SVM builds a hyperplane that maximizes the separation margin between malware and benign classes, effective for high-dimensional data using a linear kernel or RBF. Meanwhile, Logistic Regression uses a logistic function to predict binary classification probabilities in a simple but effective way as a baseline.

2.4. Evaluation Metrics

The four models—Hybrid CNN-BiLSTM, CNN-only, SVM, and Logistic Regression—were trained using datasets that had undergone balancing processes such as oversampling to avoid class imbalance that could impact performance. Cross-validation techniques were used as a validation method to ensure better model generalization and reduce the risk of overfitting. The data was divided into multiple folds (e.g., 5-fold), then each model was trained and tested in turn on different folds to ensure more robust and reliable performance evaluation.

Performance evaluation was conducted using several key metrics such as accuracy, precision, recall, and F1-score to provide a comprehensive overview of each model's classification capabilities. Experimental results showed that the Hybrid CNN-BiLSTM model delivered superior performance due to its ability to combine deep spatial feature extraction with bidirectional temporal context understanding, while the CNN-only, SVM, and Logistic Regression models served as baseline comparisons, with lower performance compared to this hybrid model.

The evaluation metrics are defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where TP (True Positive) denotes malware correctly classified as malware, TN (True Negative) denotes benign traffic correctly classified as benign, FP (False Positive) denotes benign traffic incorrectly classified as malware, and FN (False Negative) denotes malware incorrectly classified as benign.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-Score} = 2 \text{ İ } \frac{\text{Precision İ Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

2.5. Research Environment

This research was conducted in a Python-based computing environment using the Google Colab platform, which provides GPUs to accelerate deep learning model training. The data used was an Android malware dataset downloaded from Kaggle, consisting of thousands of samples with features such as permissions, API calls,

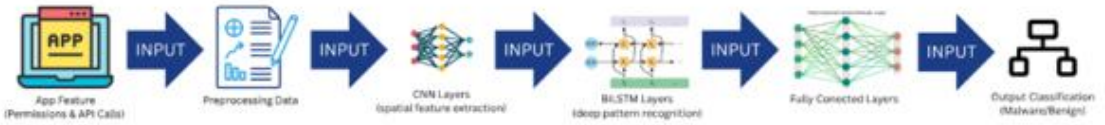

Figure 2: Design of Malware Detection Using CNN and BiLSTM

and other attributes relevant to detecting malware or benign applications. Model implementation was carried out using popular machine learning and deep learning libraries such as TensorFlow/Keras for the Hybrid CNN-BiLSTM and CNN models, and scikit-learn for the SVM and Logistic Regression models. To address class imbalance in the data, an oversampling method was used to ensure more balanced and fair model training.

As a performance validation method, a cross-validation technique for example, 5-fold was used, where the dataset is divided into several parts (folds) to ensure thorough model evaluation and reduce the risk of overfitting. This method provides more stable and reliable evaluation results when measuring metrics such as accuracy, precision, recall, and F1-score. Testing was conducted comparatively between four models: Hybrid CNN-BiLSTM, CNN alone, Support Vector Machine (SVM), and Logistic Regression, to determine the advantages and disadvantages of each in the context of efficient and accurate Android malware detection.

3. Research Results

In the evaluation phase, the performance of the four models—Hybrid CNN-BiLSTM, CNN, SVM, and Logistic Regression—was assessed using key metrics: accuracy, precision, recall, and F1-score. Prior to evaluation, the training data in each fold was balanced using the Synthetic Minority Over-sampling Technique (SMOTE) to address the significant class imbalance present in the TUANDROMD dataset. This balancing ensured that minority benign samples were adequately represented during model training, mitigating bias toward the malware class.

3.1. Data Balancing Techniques using SMOTE

The TUANDROMD dataset initially exhibited a severe class imbalance, consisting of 3,565 malware samples and only 899 benign samples. This imbalance posed a risk of bias in the learning process, as models would tend to predict the majority malware class while underperforming on benign detection. To address this, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to generate synthetic benign samples by interpolating between existing minority class instances in the feature space. After applying SMOTE, the dataset became balanced with 5,700 total samples, equally distributed into 2,850 malware and 2,850 benign instances. This balanced dataset was then used within each fold of the 5-fold cross-validation, ensuring fair representation and improved model generalization across all evaluation metrics.

```
[ ] # Save original shape
    original_shape = X_train.shape  # (samples, timesteps, features)

    # Flatten the 3D data to 2D for SMOTE
    X_train_flat = X_train.reshape((X_train.shape[0], -1))

    # Apply SMOTE on 2D data
    sm = SMOTE(random_state=42)
    X_train_res_flat, y_train_res = sm.fit_resample(X_train_flat, y_train)

    # Reshape back to 3D for CNN/LSTM
    X_train_res = X_train_res_flat.reshape((-1, original_shape[1], original_shape[2]))

    print(f"Before SMOTE: {X_train.shape}, After SMOTE: {X_train_res.shape}")

→ Before SMOTE: (3571, 241, 1), After SMOTE: (5700, 241, 1)
```

Figure 3: Code Snippet Balancing Data Using SMOTE

3.2. Average Evaluation Metrics from Cross-Validation Results

The average performance of the CNN + BiLSTM hybrid model during the 5-fold cross-validation process shows very consistent and reliable results. The average accuracy value achieved is 99.04%, which is also in line with the average precision, recall, and F1-score of 99.04% respectively. This confirms the model's ability to classify the data very precisely consistently across folds. In addition, in terms of time efficiency, the average execution time per fold was recorded at 242.42 seconds, with the overall total time during training of all folds reaching 1,212.08 seconds. The relatively short execution time indicates that the model not only excels in terms of accuracy and precision, but also in terms of computational efficiency, making it a practical and effective solution to the classification problem tested.

```
Average Accuracy: 0.9904
Average Precision: 0.9904
Average Recall: 0.9904
Average F1-Score: 0.9904

Rata-rata waktu eksekusi per fold: 242.42 detik
Total waktu eksekusi semua fold: 1212.08 detik
```

Figure 4: The Average Performance of the CNN + BiLSTM Hybrid

The evaluation results of the Convolutional Neural Network (CNN) model combined with the Synthetic Minority Over-sampling Technique (SMOTE) and cross-validation method show excellent performance with an

average accuracy of 0.9901. The average precision value of 0.9902 indicates that the model has a high ability to accurately identify positive classes without producing many false positives. In addition, the average recall of 0.9901 demonstrates the model's ability to capture almost all positive class instances, which is crucial in the context of anomaly detection where missed detections (false negatives) can have severe consequences. The F1 score of nearly 0.9901 confirms the optimal balance between precision and recall in the model, indicating the stability and reliability of predictions across various data subsets during the cross-validation process. The average execution time for each fold is estimated at 51.08 seconds, reflecting the efficiency of the training process despite involving data augmentation via SMOTE and repeated cross-validation. The total execution time of 255.41 seconds indicates that the integration of oversampling and cross-validation techniques can be executed with reasonable computational resources without compromising performance.

```
=== Cross-Validation Summary ===
Avg Accuracy : 0.9901
Avg Precision: 0.9902
Avg Recall   : 0.9901
Avg F1 Score : 0.9901

Avg Fold Execution Time: 51.08 detik
Total Execution Time: 255.41 seconds
```

Figure 5: The Average Performance of the CNN

The evaluation results of the Support Vector Machine (SVM) model applied with SMOTE oversampling and cross-validation methods demonstrated excellent classification performance, with high average evaluation metric scores. The cross-validation process yielded an average accuracy of 98.79%, indicating that the model could classify benign and malware data with nearly 99% accuracy. The very high precision of 99.77% indicates that 99.77% of all positive malware predictions made by the model were malware, indicating minimal false positives. Furthermore, the recall of 98.71% indicates that the model was able to detect 98.71% of all actual malware samples, demonstrating high sensitivity in identifying malware threats. This combination of precision and recall is reflected in an F1 score of 99.24%, demonstrating an optimal balance between the model's ability to detect malware and avoid unnecessary misclassifications. Furthermore, the average execution time of 2.5274 seconds demonstrates good computational efficiency for practical use, with a relatively short duration for the training and validation processes in each fold in cross-validation. These results illustrate that the SVM model combined with the SMOTE technique is effective in addressing class imbalance issues and can produce accurate and consistent predictions on various subsets of training data.

```
SVM Cross-Validation Average:
Accuracy      : 0.9879
Precision     : 0.9977
Recall        : 0.9871
F1 Score      : 0.9924
Execution Time: 2.5274 seconds
```

Figure 6: The Average Performance of the SVM

The overall performance evaluation of the Logistic Regression model, integrated with SMOTE and validated through cross-validation over five folds, demonstrates robust and consistent classification capabilities in the task of malware detection. The average metrics across all folds indicate an accuracy of approximately 98.6%, precision around 99.3%, recall near 98.6%, and an F1 Score of approximately 99.0%. These values reflect a highly balanced model that excels both in identifying true positives (malware correctly detected) and minimizing false positives (benign instances incorrectly classified as malware). Specifically, the averaged accuracy underscores the model's overall correctness in classification decisions, while the precision metric reveals that almost all malware predictions are correct, indicating low type I error. Recall's high average value shows the model's effectiveness in detecting malware, crucial for minimizing false negatives in security applications. The F1 Score, harmonizing precision and recall, solidifies the model's overall performance as highly reliable and effective.

```
LogisticRegression Cross-Validation Average:
Accuracy      : 0.9863
Precision     : 0.9938
Recall        : 0.9891
F1 Score      : 0.9914
Execution Time: 0.5869 seconds
```

Figure 7: The Average Performance of the Logistic Regression

The 5-fold cross-validation results reveal that the Hybrid CNN (CNN+BiLSTM) achieved the highest average performance, with accuracy, precision, recall, and F1-score all reaching 99.04%. This performance surpasses the standalone CNN model (99.01%) and significantly outperforms traditional machine learning methods, such as Logistic Regression (98.32%) and SVM (98.43%).

The performance of the Hybrid CNN model stems from combining CNN's strength in extracting spatial features with BiLSTM's ability to capture temporal dependencies between features. This synergy enables the model to learn complex behavioral patterns of Android malware, which is consistent with findings by Thakur et al. (2024), who reported that integrating CNN and LSTM architectures improves malware classification accuracy by 2–3% compared to CNN alone. Machine learning models such as Logistic Regression and SVM remain competitive but fall slightly behind due to

Table 1
Comparison Based on the Result Average Training Dataset

Method	Accuracy	Precision	Recall	F1-Score
Hybrid CNN (CNN+BiLSTM)	99.04%	99.04%	99.04%	99.04%
CNN	99.01%	99.02%	99.01%	99.01%
Logistic Regression	98.32%	99.26%	98.63%	98.94%
SVM	98.43%	99.75%	98.28%	99.01%

their limited capability in modeling sequential feature relationships.

3.3. Evaluation Metrics Final Results

The final evaluation of the hybrid model combining Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) with Synthetic Minority Over-sampling Technique (SMOTE) and cross-validation scheme on 20% of the test data demonstrated very high and consistent classification performance. The model achieved an accuracy of 0.9944, indicating optimal accuracy of the model's predictions against the actual labels. The nearly identical precision (0.9945) and recall (0.9944) demonstrated the model's ability to identify the positive class with high accuracy without producing many false positives or false negatives, which is crucial in the context of anomaly detection to minimize critical misclassifications.

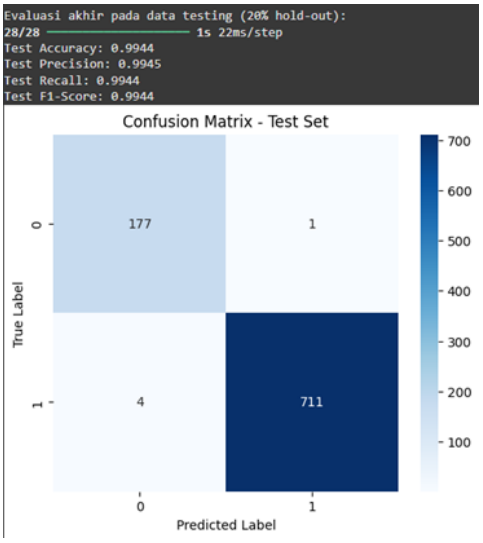

Figure 8: Confusion Matrix – Data Test and Model Hybrid CNN+BiLSTM

The F1 score of 0.9944 underscored the optimal balance between precision and recall, a vital metric for assessing model quality when dealing with class-imbalanced data. The confusion matrix showed a distribution of prediction results with 177 true negatives, 711 true positives, and only 1 false positive and 4 false negatives, further strengthening the model's reliability and validity in classification. The average execution time per step was recorded at approximately 1 second 22 milliseconds, reflecting the efficiency of the hybrid model's inference process.

The final evaluation of the Convolutional Neural Network (CNN) model equipped with the Synthetic Minority Over-sampling Technique (SMOTE) and a cross-validation scheme on 20% of the test data showed high and consistent performance. The model achieved an accuracy of 0.9910, indicating that the model's prediction accuracy relative to the actual labels is at an optimal level. Both precision and recall of 0.9910 confirm that the model not only accurately identifies positive instances (anomaly class) with minimal false positives but also captures nearly all positive instances without significant false negatives. The F1 Score of 0.9910 demonstrates a harmonious balance between precision and recall, which is an important metric in the context of anomaly detection to avoid detrimental trade-offs between these two metrics. The confusion matrix shows the distribution of predictions with 174 true negatives, 711 true positives, and only 4 false positives and 4 false negatives, further reinforcing the validity and reliability of the model in performing classification.

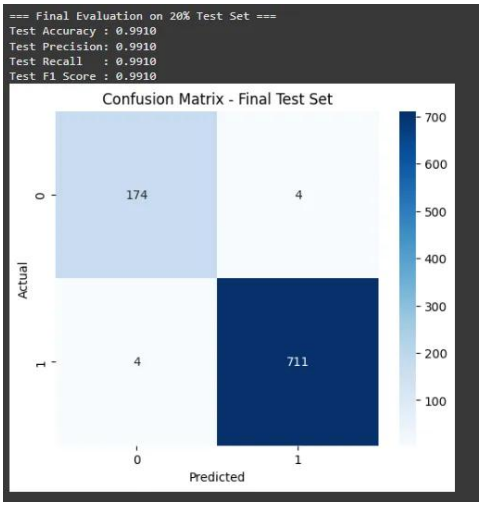

Figure 9: Confusion Matrix – Data Test and Model CNN

The final evaluation of the SVM classification model demonstrates excellent detection capability for distinguishing between benign and malware samples. The model achieved an overall accuracy of 99.55%, correctly classifying 889 out of 893 instances in the test set, indicating a negligible error rate. For class-specific performance, the precision of 100% indicates

Table 2
Comparison Based on the Result Final Evaluation (20% Data Test)

Method	Accuracy	Precision	Recall	F1-Score
Hybrid CNN (CNN+BiLSTM)	99.44%	99.45%	99.44%	99.44%
CNN	99.10%	99.10%	99.10%	99.10%
Logistic Regression	98.99%	99.16%	99.58%	99.37%
SVM	99.55%	100%	99.44%	99.72%

that every sample predicted as malware was indeed malware, eliminating false positives. The recall of 99.44% shows that 99.44% of all actual malware instances were correctly identified, with only four malware samples misclassified as benign. The F1-score of 99.72%, representing the harmonic mean of precision and recall, confirms the model's strong balance between detection sensitivity and specificity. The confusion matrix further highlights this performance: all 180 benign samples were correctly classified, while 709 out of 713 malware samples were correctly identified, resulting in only four misclassifications of malware as benign and zero misclassifications of benign as malware.

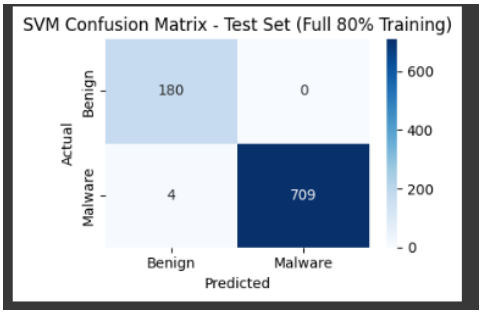

Figure 10: Confusion Matrix – Data Test and Model SVM

The final evaluation using the hold-out 20% test set demonstrated that the Logistic Regression model delivered strong classification performance in distinguishing benign and malware samples. The model achieved an overall accuracy of 98.99%, indicating that nearly all predictions were correct across both classes. Class-specific metrics further highlight this performance: a precision of 99.16% reflects the model's ability to minimize false positives, ensuring that benign applications are rarely misclassified as malware. The recall of 99.58% demonstrates the model's high sensitivity in detecting malware samples, with only three malware instances misclassified as benign. The resulting F1-score of 99.37% confirms the excellent balance between precision and recall. Analysis of the confusion matrix reveals that out of 180 benign samples, the model correctly classified 174 samples and misclassified 6 samples as malware. For the 713 malware samples, 710 were correctly identified, with only 3 false negatives.

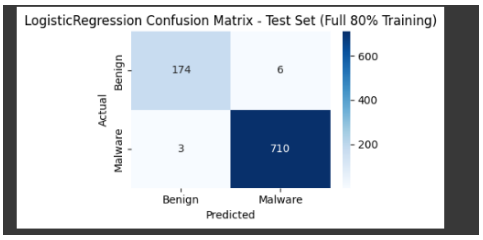

Figure 11: Confusion Matrix – Data Test and Model Logistic Regression

The final evaluation using the 20% hold-out test set confirms the stability and generalization capability of the models. The Hybrid CNN again recorded excellent results, achieving 99.44% accuracy and 99.44% F1-score, closely aligning with its cross-validation performance and indicating minimal overfitting.

Interestingly, the SVM model achieved the highest single-test accuracy at 99.55% and an F1-score of 99.72%, slightly surpassing the Hybrid CNN. This finding aligns with Thakur et al. (2024), who noted that SVM can perform exceptionally well when datasets are properly normalized and balanced via oversampling, as SVM leverages margin optimization between classes effectively. The standalone CNN remained stable with approximately 99.10% accuracy, while Logistic Regression reached around 98.99% accuracy. The small gap (less than 0.5%) between cross-validation and final evaluation results suggests robust model stability and reproducibility.

4. Conclusions and Recommendations

4.1. Conclusions

This study aimed to develop and evaluate a Hybrid CNN-BiLSTM model as a novel approach for Android malware detection. The proposed architecture leverages the strength of Convolutional Neural Networks (CNN) for extracting spatial features from static Android application data and combines it with Bidirectional Long Short-Term Memory (BiLSTM) networks to capture complex sequential and temporal dependencies. The experiments were conducted using the TUANDROMD dataset, a publicly available Android malware dataset, with the incorporation of Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance. Through five-fold cross-validation, the Hybrid CNN-BiLSTM model consistently outperformed conventional machine learning methods, including standalone

CNN, Support Vector Machine (SVM), and Logistic Regression. The model achieved near-perfect scores in terms of accuracy, precision, recall, and F1-score across all folds. Moreover, this superior performance was confirmed on a 20% hold-out test set, indicating the model's robustness and generalization ability.

These findings highlight that the integration of CNN and BiLSTM significantly enhances feature representation and reduces misclassification errors compared to traditional approaches. The key contributions of this research include the implementation of a Hybrid CNN-BiLSTM architecture on the TUANDROMD dataset for Android malware detection, a comprehensive comparison with baseline models such as CNN, SVM, and Logistic Regression to demonstrate the advantages of the proposed hybrid model, and the introduction of SMOTE and cross-validation strategies to ensure reliable performance evaluation.

The implications of these findings suggest that hybrid architectures can play a vital role in addressing the challenges of malware evolution and sophisticated obfuscation techniques. This approach offers a strong foundation for future deep learning-based mobile security systems and can be extended to Internet of Things (IoT) environments where similar security concerns are prevalent. However, the study has several limitations. The evaluation was limited to a single dataset and feature set, which may not fully represent the diversity of real-world Android applications or malware behaviors. Additionally, computational constraints on the Google Colab platform restricted extensive hyperparameter tuning and experimentation on larger-scale datasets. In conclusion, this research successfully demonstrates that the Hybrid CNN-BiLSTM model delivers significant improvements in Android malware detection compared to conventional machine learning approaches, thereby contributing to the advancement of adaptive and accurate mobile security solutions.

4.2. Recommendations

Building upon the findings of this study, several directions for future research are proposed to enhance the effectiveness, generalization, and scalability of the Hybrid CNN-BiLSTM model for Android malware detection:

- **Exploration of Alternative Hybrid Architectures.** While this research focused solely on the Hybrid CNN-BiLSTM architecture, future studies should explore other combinations such as CNN-GRU, CNN-Transformer, or Attention-based CNN-BiLSTM. These alternatives may better capture long-term feature dependencies and complex temporal patterns, potentially improving classification performance.
- **Utilization of Larger and More Diverse Datasets.** The TUANDROMD dataset used in this study is limited in terms of sample size and malware diversity. Future research could incorporate larger and heterogeneous datasets, such as Drebin, AndroZoo, or combined datasets from multiple repositories. This would enhance the model's robustness and enable evaluation against emerging malware variants in real-world conditions.
- **Real-Time Implementation and Evaluation on Android Devices.** The current experiments were conducted offline using static datasets. Future work should include real-time deployment and testing on Android devices or emulators, focusing on computational efficiency, memory usage, and inference time. Such evaluation is crucial to assess the feasibility of integrating the model into mobile security solutions.

References

Alhussen, A. (2024). Advanced Android malware detection through deep learning optimization. *Engineering, Technology and Applied Science Research*, 14(3):14552–14557.

Almuqren, A., Frikha, M., and Albuali, A. (2023). Automated malware detection based on a machine learning algorithm. In *2023 IEEE 10th International Conference on Communications and Networking, ComNet 2023 – Proceedings*.

Applications, A. A. (2024). Harsenning deep learning techniques for predictive cybersecurity: Challenges and solutions. *International Journal of Innovative Research in Advanced Engineering*, 11:723–728.

Devidas Misalkar, H. and Harshavardhanan, P. (2024). Assessing the efficacy of machine learning classifier for Android malware detection. *Pg 1 J. Integr. Sci. Technol*, 2024(4).

Gao, C., Wu, B., Huang, G., Wu, Y., Li, H., and Yuan, W. (2024). A comprehensive study of learning-based Android malware detectors under challenging environments. In *Proceedings – International Conference on Software Engineering*.

Guerra-Manzanares, A. (2024). Machine learning for Android malware detection: Mission accomplished? A comprehensive review of open challenges and future perspectives. *Computers & Security*, 138.

Jouhari, M. and Guizani, M. (2024). Lightweight CNN-BiLSTM based intrusion detection systems for resource-constrained IoT devices. n.d.

Kotadia, A., Masalia, B., Mehra, O., and Pathak, L. (2024). Machine learning for threat detection in softwares. *International Journal of Innovative Science and Research Technology (IJISRT)*, pages 2402–2413.

Liu, L., Wang, B., Yu, B., and Zhong, Q. (2017). Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology and Electronic Engineering*, 18(9):1336–1347.

Lu, T., Du, Y., Ouyang, L., Chen, Q., and Wang, X. (2020). Android malware detection based on a hybrid deep learning model. *Security and Communication Networks*, 2020.

M, M. Y. (2024). A new malware classification framework based on deep learning algorithms. *International Journal of Scientific Research in Engineering and Management*.

Meijin, L., Zhiyang, F., Junfeng, W., Luyu, C., Qi, Z., Tao, Y., Yinwei, W., and Jiaxuan, G. (2022). A systematic overview of Android malware detection. *Applied Artificial Intelligence*, 36(1).

N, D. and J, N. (2023). Review on malware classification with a hybrid deep learning. *Journal of IoT and Machine Learning*.

Osamor, F. and Wellman, B. (2023). A deep learning-based hybrid model for optimal anomaly detection. In *Proceedings – 2023 Congress in Computer Science, Computer Engineering, and Applied Computing, CSCE 2023*, pages 650–656.

Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., and Xiang, Y. (2021). A survey of Android malware detection with deep neural models. *ACM Computing Surveys*, 53(6).

Rana, K., Gupta, S., Kaur, G., and Yadav, A. L. (2024). Malware detection in network traffic using machine learning. In *Proceedings of the 3rd International Conference on Applied Artificial Intelligence and Computing, ICAAIC 2024*, pages 358–362.

Rathore, H., Agarwal, S., Sahay, S. K., and Sewak, M. (2019). Malware detection using machine learning and deep learning. In *Communications in Computer and Information Science*, pages 1–28. Springer.

Selvaganapathy, S. G., Sadasivam, S., and Ravi, V. (2021). A review on Android malware: Attacks, countermeasures and challenges ahead. *Journal of Cyber Security and Mobility*, 10(1):177–230.

Sem, t. (2024). Cyber threat detection using machine learning. *International Journal of Scientific Research in Engineering and Management*.

Shah, A. and Nawaf, L. (2024a). Malware detection using deep learning approaches.

Shah, A. and Nawaf, L. (2024b). Malware detection using deep learning approaches.

Swathi, R., Depuru, S., Sakthivel, M., Sivanantham, S., Amala, K., and Ande, P. K. (2024). A hybrid malware detection system for enhanced cloud security utilizing trust-based glowworm swarm optimization and recurrent deep neural networks. *Communications on Applied Nonlinear Analysis*, 31(5s).

Thakur, P., Kansal, V., and Rishiwal, V. (2024). Hybrid deep learning approach based on LSTM and CNN for malware detection. *Wireless Personal Communications*, 136(3):1879–1901.

Wajahat, A., He, J., Zhu, N., Mahmood, T., Saba, T., Khan, A. R., and Alamri, F. S. (2024). Outsmarting Android malware with cutting-edge feature engineering and machine learning techniques. *Computers, Materials and Continua*, 79(1):651–673.

Yang, J. and Gui, C. (2024). Android malware detection framework based on sensitive opcodes and deep reinforcement learning. *Journal of Intelligent and Fuzzy Systems*, 46(4).

Yoo, S., Kim, S., Kim, S., and Kang, B. B. (2021). AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. *Information Sciences*, 546:420–435.