Contents lists available at JESA

Journal of Engineering and Science Application

journal homepage: https://jesa.aks.or.id/index.php/jesa/index

A Robust Hybrid Approach for Malware Detection: Leveraging CNN and LSTM for Encrypted Traffic Analysis

Arif Mudi Priyatno[a], Yunia Ningsih[b], Arnes Yuli Vandika[c], Muhammadong[a]

[a]*Department of Digital Business, Universitas Pahlawan Tuanku Tambusai, Riau, Indonesia*
[b]*Department of informatics engineering, Faculty of industry, Universitas Trisakti, Jakarta, Indonesia*
[c]*Universitas Bandar Lampung, Lampung, Indonesia*
[d]*Universitas Negeri Makassar, Sulawesi Selatan, Indonesia*

Article Info

Keywords:

Encrypted Traffic Classification;
Hybrid CNN-LSTM Detection;
Deep Learning Malware Analysis

ABSTRACT

The rapid growth in Internet usage and advancements in network technologies have escalated the risk of network attacks. As the adoption of encryption protocols increases, so does the difficulty in identifying malware within encrypted traffic. Malware represents a significant danger in cyberspace, as it compromises personal data and harms computer systems. Network attacks involve unauthorized access to networks, often aiming to disrupt or damage them, with potentially severe consequences. To counter these threats, researchers, developers, and security experts are constantly innovating new malware detection techniques. Recently, deep learning has gained traction in network security and intrusion detection systems (IDSs), with models such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) showing promise in detecting malicious traffic. Despite these advancements, extracting relevant features from diverse malware types remains a challenge. Current solutions demand substantial computational resources and are often inefficient for large datasets. Additionally, existing image-based feature extraction methods consume significant resources. This study tackles these issues by employing a 1D CNN alongside LSTM for the detection and classification of malicious encrypted traffic. Using the Malware Analysis benchmark dataset, which consists of 42,797 malware and 1,079 goodware API call sequences, the proposed model achieved an accuracy of 99.2%, surpassing other state-of-the-art models.

Corresponding Author:

Arif Mudi Priyatno
Department of Digital Business
University of Pahlawan Tuanku Tambusai
Email: arifmudi@universitaspahlawan.ac.id

1. INTRODUCTION

In the modern era, computer networks serve as a crucial infrastructure, drawing increased focus on security concerns [1]. Encryption technology has played a pivotal role in preserving users' privacy, autonomy, and anonymity on the Internet. However, it has also inadvertently enabled malicious actors to

evade anomaly detection systems [2]. For example, attackers can infiltrate systems by encrypting malware traffic, while tools like Tor facilitate illegal activities in the darknet [3]. This misuse of encryption poses major challenges to detect network anomalies and securing system management. Therefore, the identification of encrypted malware has become a priority for both academic researchers and industry practitioners. Additionally, the rise of artificial intelligence (AI) has opened new avenues for creating intelligent systems capable of making autonomous, real-time decisions [4].

Malware, which refers to harmful software designed to damage devices or compromise data, has evolved in sophistication, leading to significant threats in cyberspace [5]. Malware attacks can disrupt organizations and individuals by corrupting files, stealing confidential data, or simply causing inconvenience. Malware is divided into different families, each displaying unique behaviors. Currently, variants of the Mirai malware, like Satori and Miori, are notably increasing network traffic directed at targeted victims, including corporate systems. Attackers often rely on botnets composed of compromised Internet of Things (IoT) devices to carry out these large-scale attacks. Consequently, detecting malicious traffic early in the malware distribution process is critical to preventing widespread infections and reducing the severity of attacks [6]. While early detection is crucial, various approaches to detecting malware have been proposed in recent years.

A biLSTM model combined with CNN was developed to classify malware in cloud systems, achieving close to 90% accuracy. However, replacing the biLSTM with a standard LSTM resulted in reduced performance in most cases [7]. Another study applied word embedding techniques, typically used in natural language processing (NLP), on opcode data from disassembled executables and achieved an average AUC of 0.99 for malware detection [8]. Similarly, a combination of LSTM and CNN was used to detect insider threats in the CERT insider threat dataset, which contained 32 million log lines, achieving an AUC of 94.49% by using LSTM to extract behavioral features and CNN for classification [9].

A hybrid CNN-SVM model experiment reported varying accuracies, with CNN-SVM at 77.22%, GRU-SVM at 84.92%, and MLP-SVM at 80.46% [10]. Additionally, CNN-LSTM hybrid models were introduced for image transformation and feature extraction, achieving accuracies between 95.5% and 96.64% [11]. Another approach combined CNN with a genetic algorithm for multi-objective optimization, resulting in 97.1% accuracy [12]. An ensemble of extreme learning machines (ELMs) with CNNs achieved 96.3% accuracy [13]. A deep learning-based hybrid visualization method, specifically for IoT systems, reached up to 98.79% accuracy [14]. Hybrid architectures like CNN-BiLSTM and CNN-BiGRU achieved accuracies ranging from 94.48% to 96.3% [15]. A dynamic deep learning system, DL-Droid, was proposed for detecting malicious Android applications, achieving a detection rate of 97.8% [16].

The Internet has become a primary enabler of modern malware attacks, and as the number of devices connected to the Internet continues to rise, securing these devices has become essential to prevent the potential loss of personal or sensitive information [17]. Over the past decade, malware has increased at an alarming rate, and no method has been entirely reliable for detecting all malware types. Advanced obfuscation and packing techniques employed by newer malware make detection extremely challenging using traditional methods [18].

Deep learning holds immense potential to revolutionize malware detection methods due to its ability to generate intricate patterns that capture the unique characteristics of specific objects by autonomously learning from large datasets [7]. Although deep learning (DL) in network security is relatively new, it has attracted significant attention due to its strong autonomous learning capabilities [19]. Additionally, the advancements and increased availability of GPU processors have significantly accelerated matrix computations and large-scale mathematical operations, further facilitating DL-based approaches. Most deep learning approaches for detecting malicious traffic are categorized by network models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), or unsupervised learning methods like autoencoders. However, current research reveals several key challenges. Many systems rely on a flow-based approach, which involves collecting and classifying packets over time before analyzing them offline. Additionally, feature extraction in flow-based detection takes a significant amount of time, which extends the detection process. Furthermore, this method requires substantial resources, including memory and storage, to process and analyze the accumulated traffic, particularly during deep inspection of aggregated data.

This study introduces a novel approach that emphasizes the development of hybrid techniques by integrating LSTM within CNN. In this method, CNN is utilized to extract malware features, followed by LSTM to classify the malware.

2. RESEARCH METHOD

This section provides a comprehensive overview of the experimental setup and evaluation processes that were implemented to gauge the performance of the proposed CNN-LSTM model. The primary objective of the experiments was to classify malware in an encrypted network environment, addressing the challenge of

detecting malicious activity within secure traffic. To build and test the model, the research leveraged Python's widely-used open-source deep learning framework, which incorporates a TensorFlow backend. TensorFlow's robust computational capabilities were instrumental in enabling efficient model development and training. The experiments were conducted using a standard personal computer outfitted with an HP Proliant DL380p Gen8 server, equipped with 20GB of RAM. This hardware provided sufficient processing power to handle the complexities of deep learning computations, ensuring that the model could be trained on large datasets without significant delays. The entire methodology is clearly outlined in Figure 1, which details each step taken in the development and integration of the embedded LSTM with a shallow Convolutional Neural Network (CNN). This figure is crucial for understanding the workflow and approach taken to achieve accurate classification results.

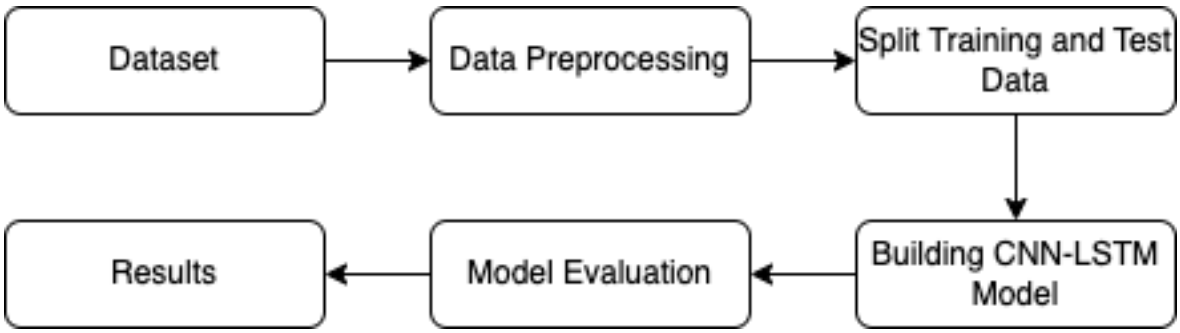


Figure 1 Proposed Methodology

2.1. Data

This study leverages a publicly accessible malware dataset, originally made available by the United States Air Force for research purposes. The availability of this dataset presents a crucial opportunity for researchers to assess progress in malware detection and classification technologies. It consists of 42,797 malware API call sequences and 1,079 sequences from goodware. Each sequence includes the first 100 unique consecutive API calls tied to the parent process, obtained from the 'call' elements in Cuckoo Sandbox reports. Figure 2 shows the distribution of classes within the dataset. To elaborate, this dataset allows for an in-depth exploration of how various models can distinguish between malicious and benign software based on their API behavior. The ability to track and analyze API calls is critical for identifying the underlying behavior of software, as malware often disguises its operations. Using this dataset offers valuable insights into real-world performance and challenges in malware detection, especially in environments where API call sequences are crucial indicators of potential threats. The well-curated data also ensures that models can be trained and validated efficiently, enabling researchers to experiment with a variety of techniques for improving detection accuracy.

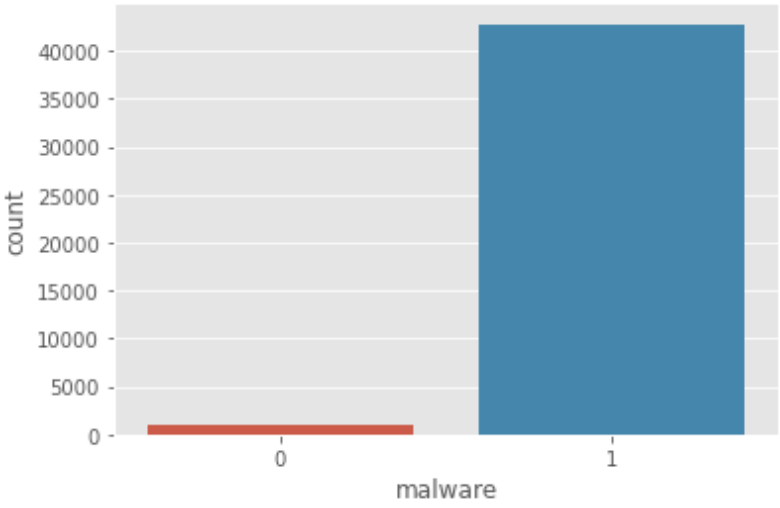


Figure 2 distribution of classes within the dataset

2.2. Data Preprocessing

In this study, the data preprocessing phase plays a crucial role in preparing the malware detection dataset for the CNN-LSTM model. The dataset consists of 42,797 malware API call sequences and 1,079 goodware sequences, each containing the first 100 unique consecutive API calls. These API calls are critical in determining the behavior of the software, especially for distinguishing between malicious and benign software. To ensure consistency and quality of the input data, the preprocessing begins with extracting the relevant API call sequences from Cuckoo Sandbox reports, associating them with their respective parent processes. This initial step allows the dataset to reflect real-world scenarios where API call behavior can indicate malicious intent.

Following the extraction, the API calls are transformed into a numerical format that the deep learning model can interpret. Normalization is applied to standardize the input values, ensuring that the data across different samples is consistent. This step helps mitigate potential biases in the input data, leading to more effective learning. Additionally, given the data balancing techniques, such as oversampling the benign samples, such as oversampling the benign class or undersampling the malware class, are likely employed. This ensures the model does not become biased towards detecting malware while neglecting benign software, improving the model's overall accuracy and robustness during training and evaluation.

2.3. Split Training and Test Data

In this research, the dataset is carefully divided into training and testing sets to ensure that the CNN-LSTM model can be evaluated effectively. Approximately 80% of the data is allocated for training, where the model learns to recognize patterns and relationships within the API call sequences. The remaining 20% of the data is set aside for testing, providing an unbiased evaluation of the model's performance on unseen data. This division allows for a comprehensive assessment of how well the model generalizes beyond the data it was trained on. By keeping the testing set separate, the study ensures that the evaluation accurately reflects the model's ability to classify malware and benign software in real-world scenarios, minimizing overfitting and enhancing the validity of the results.

2.4. CNN-LSTM model

The architecture of the CNN-LSTM model combines the strengths of both Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for effective malware detection in encrypted network traffic. The input to the model consists of raw network traffic data, which undergoes an embedding process in the first layer. The purpose of the embedding layer is to convert the raw input data into dense vectors, allowing the model to understand and work with the features more effectively. A dropout layer follows this embedding to reduce overfitting by randomly dropping units during training, ensuring that the model does not become too specialized to the training data.

The embedded data is then passed through a 1D Convolutional Neural Network (Conv1D) layer, which acts as a feature extractor. This layer is responsible for learning spatial hierarchies and detecting patterns within the network traffic data. By applying filters across the input, the Conv1D layer highlights significant patterns that may indicate malicious activity. After the convolutional operation, batch normalization is applied to normalize the data, improving the model's convergence speed and stabilizing the learning process. MaxPooling, another key layer, then reduces the dimensionality of the data by selecting the most important features, which helps the model focus on the most critical aspects of the input without being overwhelmed by noise.

After feature extraction and dimensionality reduction, the output is processed by an LSTM layer. The LSTM is particularly useful in this architecture because it excels at handling sequential data, which is crucial for detecting temporal patterns in network traffic, such as API call sequences. The LSTM layer analyzes the time-based relationships and dependencies between features, which is essential for distinguishing between benign and malicious behaviors. This temporal analysis is enhanced by another dropout layer, which again helps reduce overfitting by adding a degree of randomness during training, making the model more robust when applied to new data.

Finally, the processed data is passed through a fully connected (dense) layer, which serves as the classification layer. This layer takes the high-level features extracted by the CNN and LSTM layers and uses them to classify the network traffic as either malicious or benign. The softmax activation function in this layer ensures that the output is a probability distribution, where the class with the highest probability is selected as the prediction. This comprehensive architecture—combining the strengths of CNN for feature extraction and LSTM for sequence modeling—enables the model to perform highly accurate malware detection within encrypted network traffic.

2.5. Evaluation

The experimental evaluation of classification algorithms was conducted using several performance metrics, including classification accuracy, specificity, sensitivity, error rate, ROC curve, and execution time. The actual and predicted classification results were organized into a confusion matrix, which is commonly used to assess the performance of a classification model on test data with known values [20]. After generating the confusion matrix for each algorithm, the metrics—accuracy, sensitivity, specificity, and error rate—were calculated using specific formulas [21].

The performance metrics used to evaluate the algorithms are detailed as follows: Accuracy, as shown in Equation 1, measures the proportion of correctly classified instances by considering true positives, true negatives, false positives, and false negatives [22]. The False Positive (FP) Rate, defined in Equation 2, indicates how often benign instances are wrongly classified as malicious, with a lower FP rate reflecting better model performance. True Positive (TP) Rate, outlined in Equation 3, refers to the proportion of malicious instances (positive samples) that are correctly classified.

Precision and Recall, key metrics that assess the proportion of correctly predicted positives, are combined into the F-measure (Equation 4), which balances these values as the harmonic mean [23]. A high F-measure is crucial as it demonstrates high precision and recall [24]. Finally, the Receiver Operating Characteristic (ROC) Curve is used to plot the true positive rate against the false positive rate, with a higher ROC value indicating superior model performance.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (1)$$

$$FPR = \frac{FP}{FP + TN} \quad (2)$$

$$TP\ Rate = \frac{TP}{TP + FN} \quad (3)$$

$$F\text{-}Measure = 2\ x\ \frac{Precision}{Precision + Recall} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

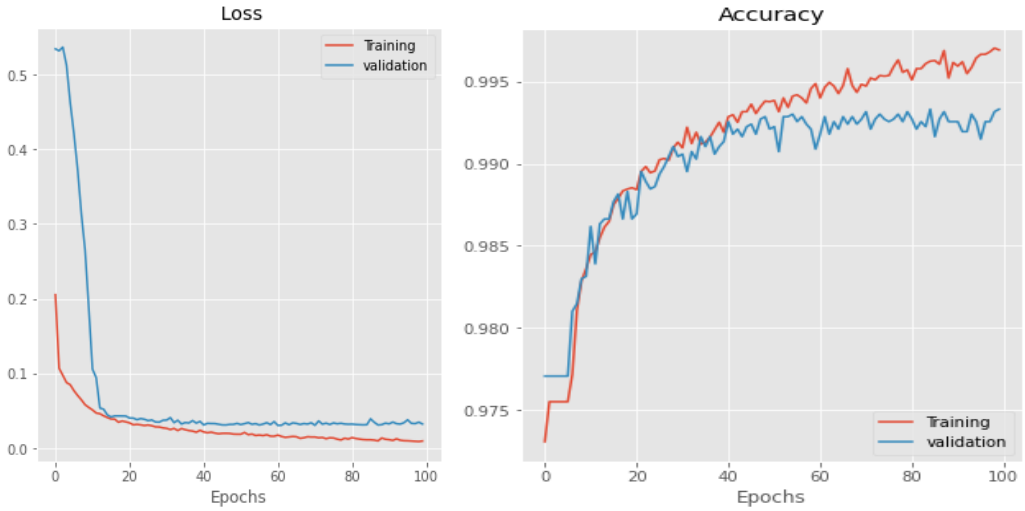
$$Recall = \frac{TP}{TP + FP} \quad (6)$$

3. RESULTS AND DISCUSSION (10 PT)

The findings in Table 1 demonstrate the strong performance of the CNN classification model for malware detection, achieving an overall accuracy of 97%. The model exhibited a perfect recall rate of 100% for detecting malware, accurately identifying all malicious instances. Additionally, the precision rate for malware was 97%, contributing to an f1-score of 97%, signifying a balanced performance between precision and recall. However, the model struggled with the benign class, showing no precision, recall, or f1-score, likely due to the dataset's imbalance, which included far fewer benign samples (283) compared to malware samples (10,686). Despite this imbalance, the weighted averages for precision, recall, and f1-score remained robust at 97%, indicating the model's effectiveness in classifying the more abundant malware instances.

Table 1 Classification Used Standard CNN

	Benign	Malware
Precision	0	97
Recall	0	100
F1-Score	0	97
Accuracy	97	



(a) (b)

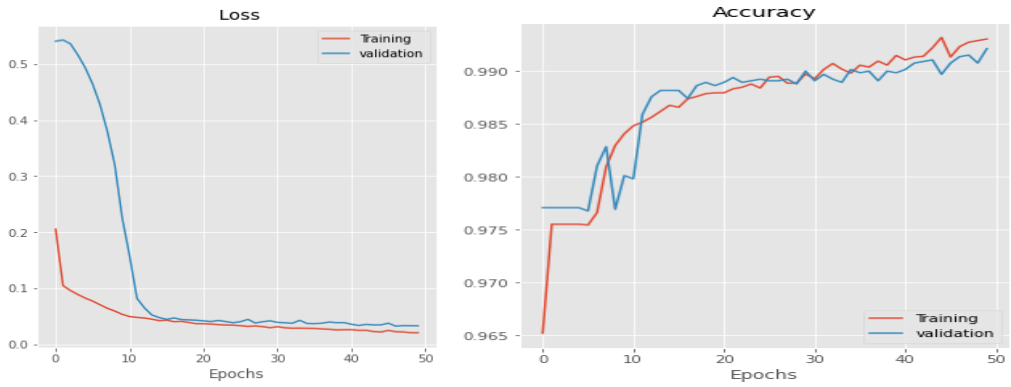
Figure 3 Training-validation (a) Loss and (b) Accuracy with standard CNN

Figure 3(a) a depicts the trend of training and validation loss across epochs. At the beginning, both losses decline rapidly; however, after around 20 epochs, the validation loss stabilizes, while the training loss continues to drop. This divergence suggests that the model may be overfitting, as it continues improving on the training data while losing generalization capability for new data. Despite the stable validation loss at a low value, this imbalance points to potential issues with the model's regularization. Figure 3b highlights the comparison between training and validation accuracy over time. Initially, both accuracies rise steadily, but a gap develops as training continues, with training accuracy reaching nearly 99% while validation accuracy stalls around 95%. This divergence between training and validation accuracy further indicates overfitting, as the model adapts too specifically to the training data and struggles to generalize effectively on unseen data.

Table 2 Classification Proposed

	Benign	Malware
Precision	89	99
Recall	65	100
F1-Score	75	99
Accuracy	99	

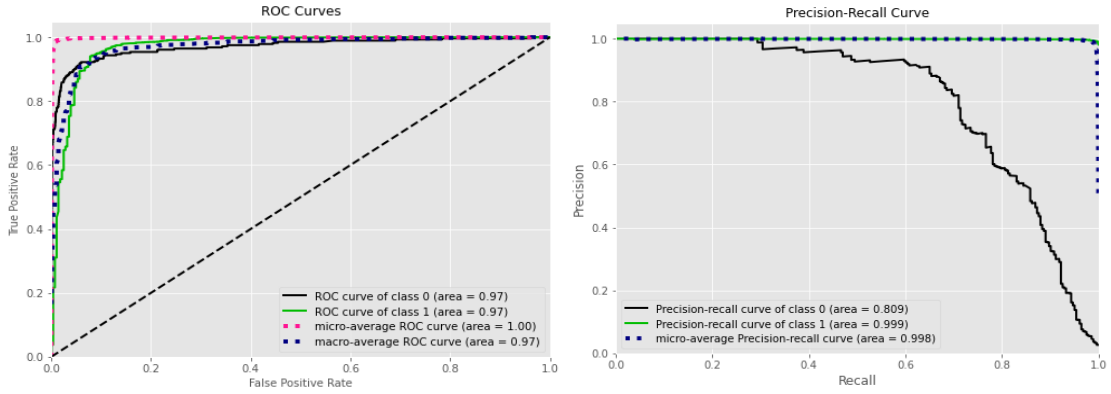
Table 2 outlines the model's precision, recall, and f1-score for both benign and malware classes, along with its overall performance metrics. For the malware class, the model shows exceptional results, achieving a precision of 99%, a recall of 100%, and an f1-score of 99%, based on a sample size of 10,686 instances. These metrics demonstrate the model's effectiveness in correctly identifying malware with near-perfect precision and recall. On the other hand, performance for the benign class, although lower, is still commendable, with a precision of 89%, a recall of 65%, and an f1-score of 75%, derived from a smaller support of 283 instances. This difference is likely due to the inherent imbalance in the dataset, where malware instances significantly outnumber benign instances, making the model more adept at malware detection. Despite this, the overall accuracy of the model stands at an impressive 99%, calculated from a total of 10,969 instances. Further analysis shows that the macro-average scores (which treat both classes equally) are 94% for precision, 83% for recall, and 87% for f1-score. These values highlight the challenge posed by the lower number of benign instances. However, the weighted averages (which give more importance to the more frequent malware class) remain remarkably high at 99% for precision, recall, and f1-score, underscoring the model's robust performance, especially in detecting malware.



(a) (b)

Figure 4 Training-validation (a) Loss and (b) Accuracy with Proposed

Figure 4a illustrates the relationship between training and validation loss over 50 epochs for the CNN-LSTM model. As training progresses, the loss for both training and validation steadily declines, indicating that the model is learning effectively by adjusting its parameters. Notably, the validation loss remains close to the training loss, suggesting good generalization with minimal overfitting. The closeness of the two curves indicates the model's stability during training, which is a positive sign of its performance, as it consistently learns without over-adapting to the training data. Figure 4b presents the comparison of training and validation accuracy across the same 50 epochs. Both training and validation accuracy show consistent improvement throughout the epochs. Initially, the validation accuracy is slightly behind the training accuracy but catches up over time, eventually fluctuating around similar values by the end of the training process. This convergence suggests that the model is not only learning efficiently from the training data but is also generalizing its knowledge effectively to the validation set. The stable high accuracy observed towards the end of the training period further highlights the model's strong performance in classifying malware and benign instances with minimal overfitting. This consistent performance, reflected in both loss and accuracy, confirms the model's robustness and its ability to maintain high accuracy for unseen data.



(a) (b)

Figure 5 Result (a) ROC and (b) Precision-recall curve with Proposed

Figure 5a shows the ROC curves for the CNN-LSTM model for each class. The ROC curve plots the true positive rate against the false positive rate to assess how well the model distinguishes between malware (class 1) and benign (class 0). The curve for malware achieves an almost perfect AUC of 100, indicating exceptional classification accuracy, while the benign class has an AUC of 97, demonstrating strong performance. The macro-average ROC curve balances these and shows an AUC of 99, highlighting the model's excellent sensitivity and specificity. The proximity of the curves to the top-left corner suggests the model effectively minimizes false positives while maintaining high detection accuracy. Figure 5b illustrates the Precision-Recall (P-R) curves for both classes. The curve for class 1 (malware) remains consistently near the top-right corner, showing that the model achieves near-perfect precision even with high recall, effectively detecting malware with minimal false positives. For class 0 (benign), however, the curve declines more sharply, indicating greater difficulty in balancing precision and recall, especially as recall increases. The macro-average P-R curve remains strong at 99, reflecting the model's overall ability to balance precision and

recall well across both classes. The lower precision for the benign class suggests the model could improve in reducing false positives for benign instances, while its malware detection remains highly reliable.

4. CONCLUSION

Based on the conducted research, the proposed hybrid CNN-LSTM model for detecting malware in encrypted network traffic has proven to be highly effective. This model addresses the key challenges in malware detection, such as the difficulty in feature extraction from various types of malware and the high resource demand for large datasets. The results demonstrate that the CNN-LSTM model can achieve an accuracy of up to 99.2%, significantly outperforming other methods used in previous studies. The model shows exceptional performance in detecting malware, with near-perfect precision and recall, particularly for the malware class. The combination of CNN for feature extraction and LSTM for classification effectively handles encrypted traffic, which is a major concern in modern malware detection. The model evaluation using various metrics, including the confusion matrix, ROC curve, and precision-recall curve, indicates that the model has a low error rate, especially in classifying malicious traffic. However, a remaining challenge is improving the model's performance in detecting benign traffic, which, although accurate, has a higher error rate compared to malware detection.

REFERENCES

[1] S. Soderi, D. Masti, and Y. Z. Lun, "Railway Cyber-Security in the Era of Interconnected Systems: A Survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 7, pp. 6764–6779, Jul. 2023, doi: 10.1109/TITS.2023.3254442.

[2] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, "A Critical Cybersecurity Analysis and Future Research Directions for the Internet of Things: A Comprehensive Review," *Sensors*, vol. 23, no. 8, p. 4117, Apr. 2023, doi: 10.3390/s23084117.

[3] J. Saleem, R. Islam, and M. Z. Islam, "Darknet Traffic Analysis: A Systematic Literature Review," *IEEE Access*, vol. 12, pp. 42423–42452, 2024, doi: 10.1109/ACCESS.2024.3373769.

[4] A. K. Tyagi and S. R. Addula, "Artificial Intelligence for Malware Analysis," in *Artificial Intelligence-Enabled Digital Twin for Smart Manufacturing*, Wiley, 2024, pp. 359–390. doi: 10.1002/9781394303601.ch17.

[5] V. Vasani, A. K. Bairwa, S. Joshi, A. Pljonkin, M. Kaur, and M. Amoon, "Comprehensive Analysis of Advanced Techniques and Vital Tools for Detecting Malware Intrusion," *Electronics*, vol. 12, no. 20, p. 4299, Oct. 2023, doi: 10.3390/electronics12204299.

[6] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms," *IEEE Access*, vol. 11, pp. 121118–121141, 2023, doi: 10.1109/ACCESS.2023.3328351.

[7] G. M. and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Comput. Sci. Rev.*, vol. 47, p. 100529, Feb. 2023, doi: 10.1016/j.cosrev.2022.100529.

[8] D. Demirci, N. Sahin, M. Sirlancis, and C. Acarturk, "Static Malware Detection Using Stacked BiLSTM and GPT-2," *IEEE Access*, vol. 10, pp. 58488–58502, 2022, doi: 10.1109/ACCESS.2022.3179384.

[9] M. Aljabri *et al.*, "Intelligent Techniques for Detecting Network Attacks: Review and Research Directions," *Sensors*, vol. 21, no. 21, p. 7070, Oct. 2021, doi: 10.3390/s21217070.

[10] M. J. Awan *et al.*, "Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention," *Electronics*, vol. 10, no. 19, p. 2444, Oct. 2021, doi: 10.3390/electronics10192444.

[11] A. M. Alnajim, S. Habib, M. Islam, R. Albelaihi, and A. Alabdulatif, "Mitigating the Risks of Malware Attacks with Deep Learning Techniques," *Electronics*, vol. 12, no. 14, p. 3166, Jul. 2023, doi: 10.3390/electronics12143166.

[12] M. N. Al-Andoli, K. S. Sim, S. C. Tan, P. Y. Goh, and C. P. Lim, "An Ensemble-Based Parallel Deep Learning Classifier With PSO-BP Optimization for Malware Detection," *IEEE Access*, vol. 11, pp. 76330–76346, 2023, doi: 10.1109/ACCESS.2023.3296789.

[13] A. Mallik, A. Khetarpal, and S. Kumar, "ConRec: malware classification using convolutional recurrence," *J. Comput. Virol. Hacking Tech.*, vol. 18, no. 4, pp. 297–313, Feb. 2022, doi: 10.1007/s11416-022-00416-3.

[14] E. U. H. Qazi, M. H. Faheem, and T. Zia, "HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System," *Appl. Sci.*, vol. 13, no. 8, p. 4921, Apr. 2023, doi: 10.3390/app13084921.

[15] A. I. A. Alzahrani, M. Ayadi, M. M. Asiri, A. Al-Rasheed, and A. Ksibi, "Detecting the Presence of

Malware and Identifying the Type of Cyber Attack Using Deep Learning and VGG-16 Techniques," *Electronics*, vol. 11, no. 22, p. 3665, Nov. 2022, doi: 10.3390/electronics11223665.

[16] A. R. Nasser, A. M. Hasan, and A. J. Humaidi, "DL-AMDet: Deep learning-based malware detector for android," *Intell. Syst. with Appl.*, vol. 21, p. 200318, Mar. 2024, doi: 10.1016/j.iswa.2023.200318.

[17] M. Schmitt, "Securing the digital world: Protecting smart infrastructures and digital industries with artificial intelligence (AI)-enabled malware and intrusion detection," *J. Ind. Inf. Integr.*, vol. 36, p. 100520, Dec. 2023, doi: 10.1016/j.jii.2023.100520.

[18] T. Muralidharan, A. Cohen, N. Gerson, and N. Nissim, "File Packing from the Malware Perspective: Techniques, Analysis Approaches, and Directions for Enhancements," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–45, May 2023, doi: 10.1145/3530810.

[19] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Deep Learning for Android Malware Defenses: A Systematic Literature Review," *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–36, Aug. 2023, doi: 10.1145/3544968.

[20] S. Sanjaya, A. M. Priyatno, F. Yanto, and I. Afrianty, "Klasifikasi Diabetik Retinopati Menggunakan Wavelet Haar dan Backpropagation Neural Network," in *Seminar Nasional Teknologi Informasi Komunikasi dan Industri (SNTIKI-10)*, 2018, pp. 77–84.

[21] A. M. Priyatno and F. I. Firmananda, "N-Gram Feature for Comparison of Machine Learning Methods on Sentiment in Financial News Headlines," *RIGGS J. Artif. Intell. Digit. Bus.*, vol. 1, no. 1, pp. 01–06, Jul. 2022, doi: 10.31004/riggs.v1i1.4.

[22] A. M. Priyatno, "Spammer Detection Based on Account, Tweet, and Community Activity on Twitter," *J. Ilmu Komput. dan Inf.*, vol. 13, no. 2, pp. 97–107, Jul. 2020, doi: 10.21609/jiki.v13i2.871.

[23] A. M. Priyatno and L. Ningsih, "TF - IDF Weighting to Detect Spammer Accounts on Twitter based on Tweets and Retweet Representation of Tweets," *Sist. J. Sist. Inf.*, vol. 11, no. 3, pp. 614–622, 2022, [Online]. Available: http://sistemasi.ftik.unisi.ac.id/index.php/stmsi/issue/view/46

[24] M. R. A. Prasetya and A. M. Priyatno, "Dice Similarity and TF-IDF for New Student Admissions Chatbot," *RIGGS J. Artif. Intell. Digit. Bus.*, vol. 1, no. 1, pp. 13–18, Jul. 2022, doi: 10.31004/riggs.v1i1.5.