ISSN 2277-2685
IJESR/April-June. 2026/ Vol-16/Issue-2/39-45
Ragi Shailaja *et. al.*, /International Journal of Engineering & Science Research
Full Length Article

Hybrid Machine Learning Model For Efficient Botnet Attack Detection In IoT Environment

Ms .Shafia Tasneem[1], Ragi Shailaja[2], Erukala Renusri[3], Kappala Sri Vidya[4]
[1]Assistant Professor; Department Of Electronics And Communication Engineering Bhoj Reddy Engineering College For Women Hyderabad India.
[2,3,4]B.Tech Students; Department Of Electronics And Communication Engineering Bhoj Reddy Engineering College For Women Hyderabad India.
Mail Id; shailajaragi57@gmail.com[2]
Accepted 03-04-2026

Abstract
The rapid proliferation of Internet of Things (IoT) devices has significantly increased the vulnerability of networks to cyber threats, particularly botnet-based attacks. These attacks exploit insecure devices to perform large-scale malicious activities such as Distributed Denial of Service (DDoS), data exfiltration, and service disruption. Detecting such attacks has become increasingly complex due to evolving malware techniques and heterogeneous IoT traffic patterns.
This study presents a hybrid deep learning-based framework for detecting botnet activity using the UNSW-NB15 dataset, addressing both binary and multiclass classification tasks. Multiple deep learning models, including Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN), and Artificial Neural Networks (ANN), were implemented and evaluated. Feature selection was performed using Mutual Information to enhance computational efficiency and model performance.
Furthermore, ensemble and hybrid architectures such as CNN+LSTM, CNN+BiLSTM+BiGRU, and CNN+LSTM+GRU were explored. Experimental results demonstrate that hybrid models outperform standalone architectures, achieving detection accuracies exceeding 97%. The findings confirm that integrating spatial and temporal learning techniques significantly improves the robustness and reliability of botnet detection systems in IoT environments.

Keywords*—IoT Security, Botnet Detection, Deep Learning, Hybrid Models, CNN, LSTM, Ensemble Learning, UNSW-NB15*

Introduction

The Internet of Things (IoT) has revolutionized modern digital ecosystems by enabling seamless communication between interconnected devices across domains such as healthcare, smart homes, and industrial automation. Despite its advantages, IoT introduces critical security challenges due to the limited computational capabilities and weak security configurations of many devices.
One of the most severe threats in IoT ecosystems is botnet attacks. A botnet is formed when multiple compromised devices are remotely controlled by attackers to perform coordinated malicious actions. These attacks often result in network congestion, service downtime, and unauthorized data access.
Traditional rule-based and signature-based detection methods are inadequate for handling the dynamic and large-scale nature of IoT traffic. As a result, intelligent techniques capable of learning complex patterns are essential for effective threat detection.

Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environments

The rapid expansion of the Internet of Things (IoT) has created a highly connected environment where billions of smart devices communicate and exchange data continuously. These devices include sensors, wearable gadgets, smart home appliances, healthcare equipment, and industrial control systems. IoT technology has improved automation, monitoring, and decision-making in various sectors such as healthcare, transportation, agriculture, and manufacturing. However, the increasing number of connected devices has also introduced serious security challenges. Many IoT devices are developed with limited processing power, low memory, and weak security mechanisms, making them vulnerable to cyber-attacks.
Among the various threats faced by IoT networks, botnet attacks are considered one of the most dangerous. A botnet is a network of infected devices that are remotely controlled by attackers without the knowledge of the device owners. These compromised devices are often used to perform malicious activities such as Distributed Denial of Service (DDoS) attacks, malware distribution, data theft, and disruption of normal network services.

Since IoT devices are widely distributed and often poorly secured, attackers can easily exploit them to create large-scale botnets capable of causing significant damage.

Detecting botnet attacks in IoT environments is a difficult task because network traffic generated by IoT devices is dynamic, diverse, and produced in large volumes. Traditional intrusion detection methods based on signatures or predefined rules are limited because they can only detect known attack patterns. They are often ineffective against new and evolving threats. Therefore, there is a strong need for intelligent and adaptive security systems that can automatically learn from network data and detect malicious behavior in real time.

Machine learning and deep learning techniques have become powerful tools for network security because they can analyze complex traffic patterns and identify hidden relationships in data. Models such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Artificial Neural Networks (ANN) have shown promising results in attack detection tasks. However, single models may suffer from certain limitations. Some models are efficient in extracting features, while others are better at learning sequential behavior. Depending on only one model may reduce detection efficiency.

To overcome these limitations, this project proposes a hybrid machine learning model that combines the strengths of multiple deep learning techniques. CNN is useful for extracting spatial features from traffic data, while LSTM and RNN are effective in learning time-based patterns. ANN provides strong classification capability. By integrating these methods, the proposed system can improve detection accuracy, reduce false alarms, and provide better resistance against sophisticated botnet attacks.

This chapter explains the architecture and workflow of the proposed hybrid model for efficient botnet attack detection in IoT environments.

System Architecture

The proposed system architecture is designed to detect botnet attacks in IoT networks through a sequence of intelligent processing stages. The first stage is data collection, where network traffic information is gathered from IoT devices, gateways, sensors, and publicly available datasets such as UNSW-NB15. This dataset contains normal as well as malicious traffic records and is widely used for intrusion detection research.

After data collection, the preprocessing stage is performed to improve data quality. In this stage, missing values are handled, duplicate records are removed, categorical values are converted into numerical form, and data normalization is applied. Preprocessing ensures that the dataset becomes suitable for model training and improves learning performance.

The processed data is then passed to the hybrid deep learning layer. In this stage, different deep learning models such as CNN, LSTM, RNN, and ANN are trained individually and in combined forms such as CNN + LSTM, CNN + GRU, CNN + BiLSTM + BiGRU, and CNN + LSTM + GRU. These hybrid models capture both spatial and temporal relationships in the traffic data, which improves detection capability.

The classification layer predicts whether the incoming traffic is normal or malicious. It can also classify different categories of attacks depending on the dataset labels. Finally, the output layer presents the results in the form of alerts, accuracy values, confusion matrices, graphs, and performance reports. This helps administrators monitor threats and take preventive actions.

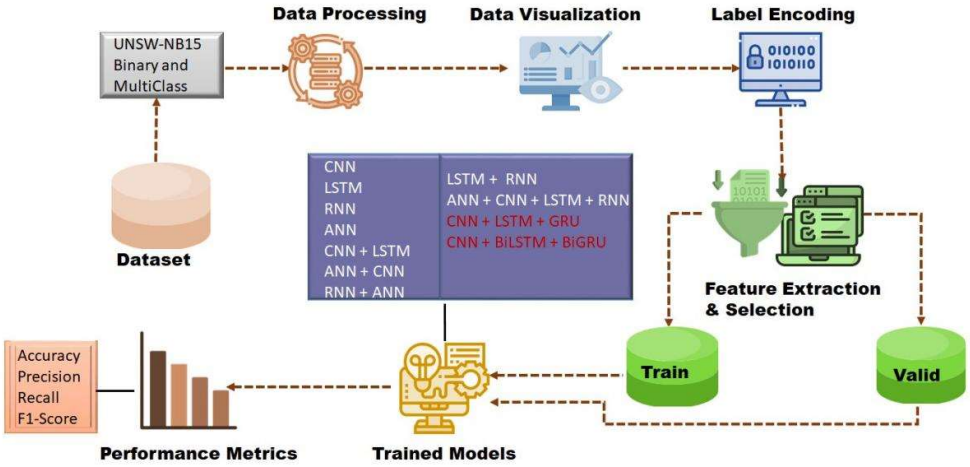

Fig 1;System Architecture

Software Requirements

This chapter describes the software requirements needed for the development and implementation of the proposed botnet attack detection system in IoT environments. The success of the system depends greatly on selecting appropriate software tools,

platforms, programming languages, and libraries that support data preprocessing, model training, testing, and performance evaluation.

The proposed system is implemented using the Python programming language because it is simple, flexible, and widely used for machine learning, data science, and artificial intelligence applications. Python provides extensive library support and allows rapid development of intelligent systems. Development platforms such as Anaconda and Jupyter Notebook are used because they offer an interactive environment for coding, executing programs, visualizing outputs, and testing machine learning models.

Several important libraries are utilized in this project. NumPy is used for numerical computation, Pandas for data handling and manipulation, Matplotlib and Seaborn for graphical visualization, Scikit-learn for preprocessing and traditional machine learning utilities, and TensorFlow or PyTorch for building advanced deep learning models.

In addition to software tools, cybersecurity concepts such as network security, cloud security, endpoint security, mobile security, application security, and IoT security are also important. These technologies help in protecting the IoT ecosystem against botnet attacks and other cyber threats.

Overall, this chapter explains the software environment, installation tools, and libraries required for implementing the proposed hybrid machine learning model for efficient botnet attack detection.

Fig 2; Types of Cyber Security

Software Environment

Cybersecurity is the practice of protecting systems, networks, devices, and digital assets from unauthorized access, attacks, and damage. As cyber threats continue to grow in frequency and sophistication, organizations require multiple layers of security to protect their infrastructure. Since IoT devices are connected to networks and exchange data continuously, cybersecurity becomes an essential part of any IoT-based solution.

Network security is one of the most important components of cybersecurity. Most cyber-attacks occur through communication networks. Network security mechanisms are designed to monitor traffic, detect suspicious behavior, and block unauthorized access. Technologies such as firewalls, intrusion detection systems, intrusion prevention systems, access control mechanisms, and threat monitoring tools are commonly used to secure networks.

Cloud security has become increasingly important because many IoT systems store and process data using cloud platforms. Cloud security protects applications, databases, virtual machines, and user data hosted in cloud environments. It uses encryption, identity management, access control, and monitoring systems to prevent attacks and data breaches.

Endpoint security focuses on securing devices such as desktops, laptops, servers, and connected endpoints. Since attackers often target user devices, endpoint security solutions such as antivirus software, ransomware protection, and endpoint detection and response tools are widely used.

Mobile security is another critical area because smartphones and tablets frequently connect to IoT platforms. Mobile devices can be exposed to phishing, malicious applications, unauthorized access, and operating system vulnerabilities. Mobile security solutions protect devices through secure authentication, device management, and malware prevention.

Zero Trust is a modern cybersecurity approach where no user or device is trusted by default. Every request is continuously verified using identity checks, monitoring, and access policies. This model is highly effective in distributed IoT environments.

The proposed botnet detection system works within this broader cybersecurity environment by adding intelligent machine learning-based attack detection capabilities.

Python Anaconda Installation

Anaconda is a popular open-source distribution used for Python programming, data science, and machine learning projects. It simplifies package management and provides pre-installed tools required for scientific computing.

During installation, advanced options may be displayed. These options include adding Anaconda to the system PATH variable and setting it as the default Python interpreter. After selecting the required options, the installation process begins and necessary packages are extracted.

Once installation is complete, the user can access tools such as Anaconda Navigator, Jupyter Notebook, Spyder IDE, and Anaconda Prompt. These tools provide a complete environment for machine learning development.

Anaconda is especially useful because it simplifies dependency management and allows quick installation of libraries such as TensorFlow, PyTorch, NumPy, and Pandas.

Libraries and Packages Used

TensorFlow is an open-source deep learning library developed by Google. It is widely used for designing neural networks, performing tensor computations, and training large-scale machine learning models. In this project, TensorFlow is used for implementing CNN, LSTM, and hybrid deep learning architectures.

Matplotlib is a visualization library used for generating charts, graphs, histograms, scatter plots, and line plots. It is used in the project for presenting model accuracy, loss curves, confusion matrices, and feature relationships.

Pandas is a powerful library for data manipulation and analysis. It provides data structures such as DataFrames that simplify loading, cleaning, filtering, sorting, and transforming datasets.

NumPy stands for Numerical Python and is used for efficient numerical operations on arrays and matrices. It supports high-speed mathematical computations and is widely used in data preprocessing and model development.

Scikit-learn is a widely used machine learning library that provides tools for classification, regression, clustering, preprocessing, feature selection, and evaluation metrics. It is used in this project for dataset splitting, label encoding, feature selection, and performance analysis.

PyTorch is another advanced deep learning framework known for flexibility and dynamic computation graphs. It supports GPU acceleration and is often used for neural network research and experimentation.

Seaborn is an additional statistical visualization library built on Matplotlib. It is useful for generating attractive correlation heatmaps, distributions, and comparative charts.

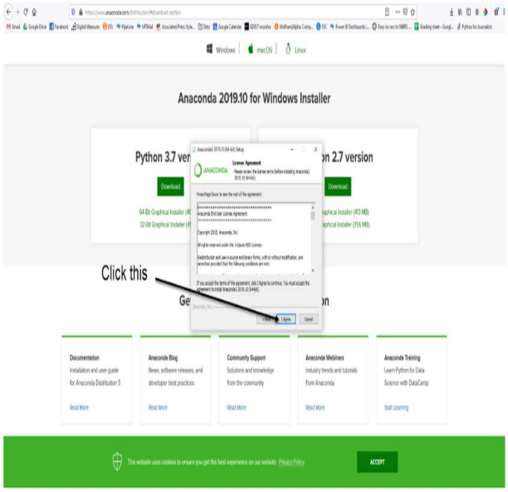

Fig 3;license Agreement

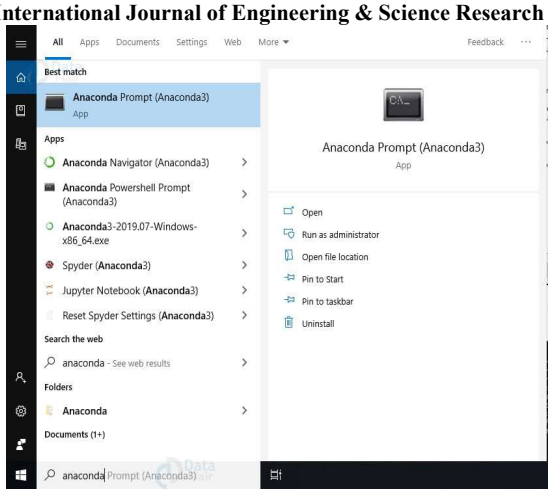

Fig 4;-Anaconda Prompt

Software Implementation

Software implementation is one of the most important phases in the development of the proposed botnet attack detection system. In this stage, the theoretical design and architecture of the system are converted into a practical and working model using suitable programming tools, frameworks, and software libraries. The implementation mainly focuses on applying machine learning and deep learning techniques to analyze IoT network traffic data and accurately detect malicious botnet activities.

The system is developed using the Python programming language because of its simplicity, flexibility, and strong support for scientific computing, data analysis, and artificial intelligence applications. Python provides a rich ecosystem of libraries that simplify model development and experimentation. The implementation environment includes Anaconda and Jupyter Notebook, which are widely used for machine learning research and development. These tools provide an interactive platform for coding, testing, visualization, and performance evaluation.

Several libraries are used in the project. NumPy is used for numerical computation, Pandas for dataset handling and preprocessing, Matplotlib and Seaborn for graphical visualization, Scikit-learn for preprocessing and feature selection, and TensorFlow or PyTorch for deep learning model implementation. These libraries make it possible to build, train, evaluate, and compare different models efficiently.

The implementation follows a systematic workflow consisting of dataset loading, preprocessing, feature selection, training, testing, and prediction. Multiple algorithms such as CNN, LSTM, RNN, and ANN are implemented individually and in hybrid combinations to improve overall detection accuracy. This chapter explains the modules, algorithms, testing procedures, and practical deployment aspects of the proposed system.

Modules

The first module of the system is data loading. In this stage, the IoT traffic dataset is imported into the Python environment. The UNSW-NB15 dataset is commonly used because it contains both normal and malicious traffic records suitable for training and evaluation.

The second module is data preprocessing. Raw datasets often contain duplicate entries, missing values, irrelevant attributes, and inconsistent formats. Therefore, preprocessing is performed to clean the data and prepare it for model training. Duplicate records are removed, null values are handled, and unnecessary columns are dropped to improve data quality.

The fifth module is feature extraction. In this stage, input variables are separated from the target class label. The selected features represent network traffic characteristics, while the output variable indicates whether the traffic is normal or malicious.

The sixth module is feature selection. Mutual Information is applied to measure the dependency between input features and the target label. Only the most relevant features are selected, reducing dimensionality and improving model efficiency.

The seventh module is data splitting. The dataset is divided into training and validation sets. The training data is used to build the model, while the validation data is used to evaluate its performance.

The eighth module is model generation. Several models such as CNN, LSTM, RNN, ANN, CNN + LSTM, ANN + CNN, RNN + ANN, LSTM + RNN, CNN + LSTM + GRU, and CNN + BiLSTM + BiGRU are trained and tested. Their performance is measured using accuracy, precision, recall, and F1-score.

The ninth module is user registration and login. A front-end interface is developed using Flask where users can securely register and access the prediction system.

The tenth module is user input and prediction. Users provide traffic parameters through the interface, and the trained model predicts whether the traffic is safe or malicious.

An extended version of the project includes advanced ensemble models such as CNN + LSTM + GRU and CNN + BiLSTM + GRU, which further improve prediction performance. The Flask-based interface also provides a user-friendly environment for testing and evaluation.

System Testing

System testing is performed after model development to ensure that the complete integrated system functions according to user requirements. It is commonly known as black-box testing because it evaluates outputs without focusing on internal code structure.

Different levels of testing are involved in the project. Unit testing checks individual modules such as data loading or prediction functions. Integration testing ensures that connected modules work together properly. System testing verifies the complete application, including user login, model prediction, and result display. Acceptance testing confirms that the system satisfies user expectations.

Several testing methods are applied. Usability testing ensures that the interface is easy to use. Load testing measures performance under heavy traffic requests. Regression testing confirms that updates do not introduce new errors. Recovery testing checks whether the system can recover after failure. Functional testing verifies whether all required operations are working correctly.

The testing process begins with planning and designing test cases for every module. These test cases are executed to observe outputs. Any defects identified are recorded and corrected. Final reports are generated to summarize system quality and readiness.

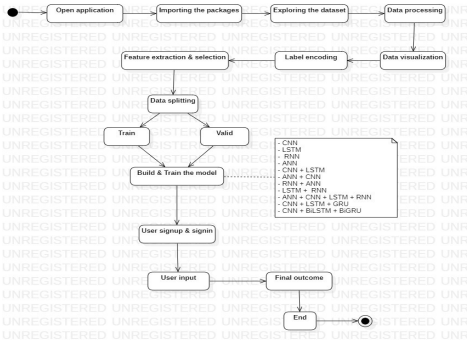

Fig 5-System Testing

Results and Discussion

Sample Outcome

The sample outcome represents the final prediction results generated after processing the network traffic dataset through the trained models. It shows whether the traffic instance is classified as normal or malicious. The output also includes evaluation measures such as accuracy, precision, recall, and F1-score. Based on the observed results, hybrid models produce more accurate predictions than individual models. The sample outcome confirms that the proposed system is capable of efficiently detecting botnet attacks in IoT environments.

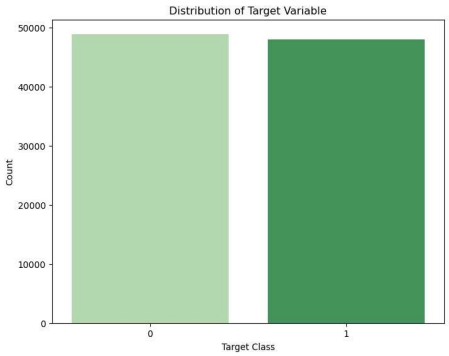

Fig 6. -Sample Outcome

Correlation Matrix

The correlation matrix illustrates the relationship between different features in the dataset. It uses color intensity to represent the strength and direction of correlation among variables. Higher positive correlation values indicate that two features increase together, while negative values indicate opposite behavior. This analysis helps identify redundant or less useful attributes. By selecting highly relevant features and removing unnecessary ones, the overall model performance improves significantly.

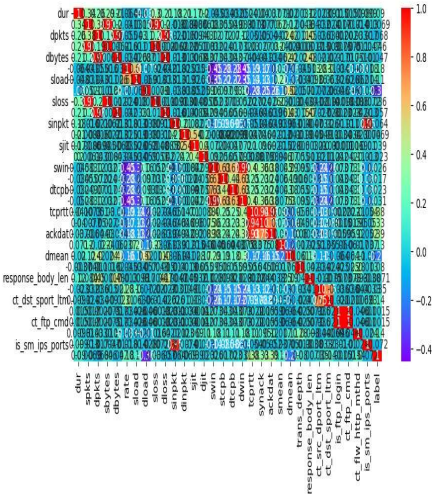

Fig 7Correlation Matrix

ANN (Artificial Neural Network)

The ANN performance graph shows the classification capability of the Artificial Neural Network model. It presents metrics such as accuracy, precision, recall, and F1-score. ANN performs well in identifying nonlinear relationships among network traffic features. However, compared to advanced deep learning and hybrid models, its performance is moderate because it may not effectively capture sequential dependencies in traffic behavior.

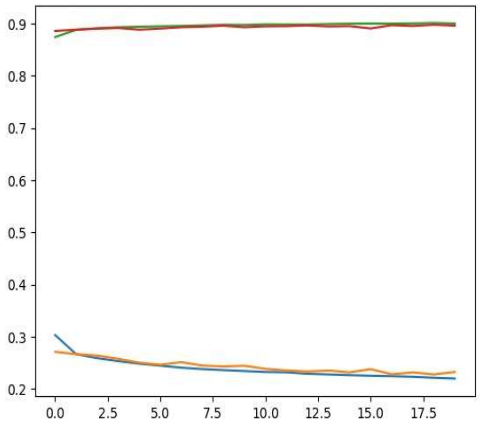

Fig 8-ANN(Artificial Neural Network

RNN (Recurrent Neural Network)

The RNN graph illustrates the results of the Recurrent Neural Network model. Since RNN is designed for sequential data processing, it is useful for analyzing time-dependent traffic patterns. The model provides satisfactory values for accuracy and recall. However, traditional RNN models may face difficulty in learning long-term dependencies due to vanishing gradient problems. Despite this limitation, RNN contributes positively when combined with other models.

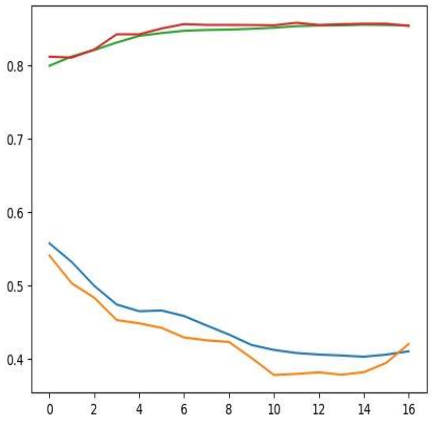

Fig 9-RNN(Recurrent Neural Network)

Conclusion

In recent years, cyber-attacks have increased significantly in both frequency and complexity, creating major challenges for modern network security systems. Among these threats, botnet attacks have emerged as one of the most serious dangers, especially in Internet of Things (IoT) environments where a large number of interconnected devices often operate with limited security protections. Compromised IoT devices can be exploited to launch Distributed Denial of Service (DDoS) attacks, spread malware, steal sensitive information, and disrupt normal network operations. Therefore, the development of intelligent and automated detection systems has become essential.

This project presented a hybrid machine learning model for efficient botnet attack detection in IoT environments. The proposed system utilized deep learning techniques such as Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Recurrent Neural Networks (RNN), Artificial Neural Networks (ANN), and advanced hybrid combinations. These models were trained and evaluated using network traffic datasets to classify normal and malicious behavior with high accuracy.

Experimental results demonstrated that hybrid models performed better than standalone algorithms. In particular, the CNN + LSTM + GRU and CNN + BiLSTM + GRU architectures achieved the highest performance, with detection accuracies exceeding 97%. These models effectively combined

feature extraction, sequential learning, and memory optimization, enabling them to identify complex traffic patterns and evolving botnet behaviors more accurately.

The evaluation metrics such as precision, recall, F1-score, and ROC-AUC further confirmed the robustness and reliability of the proposed system. The use of feature selection techniques also improved model efficiency by reducing unnecessary attributes and enhancing training performance.

Overall, the results prove that hybrid deep learning approaches are highly effective for securing IoT networks against botnet attacks. The proposed system offers a scalable, intelligent, and reliable solution capable of adapting to changing cyber threats while improving the safety of connected devices and communication networks.

Future Scope

The future scope of the proposed hybrid machine learning model is extensive and highly promising. As IoT technology continues to expand, the need for stronger and more adaptive cybersecurity systems will also increase. Several enhancements can be made to improve the performance, efficiency, and practical deployment of the proposed model.

One important direction is the integration of more advanced deep learning architectures such as Transformers, attention-based networks, and graph neural networks. These techniques may further improve the detection of complex and distributed attack patterns in large IoT ecosystems.

The system can also be optimized for real-time implementation on edge and fog computing devices. Deploying the detection model closer to IoT devices will reduce response time, lower latency, and decrease dependency on centralized cloud servers.

Another important research area is encrypted traffic analysis. Future versions of the model can be designed to detect malicious behavior in encrypted communications without violating user privacy or compromising secure protocols.

Adaptive and self-learning mechanisms can be introduced so that the model continuously updates itself using new threat intelligence. This will make the system more effective against zero-day attacks and newly emerging malware variants.

The integration of blockchain technology can strengthen trust, transparency, and secure communication among IoT devices. Blockchain-based identity management and decentralized logging can improve overall network security.

Future work may also focus on developing lightweight deep learning models suitable for low-power and resource-constrained IoT devices. Techniques such as model pruning, quantization, and knowledge distillation can help reduce memory and computational requirements.

Explainable Artificial Intelligence (XAI) can be incorporated to improve transparency and help administrators understand why certain traffic is classified as malicious. This will increase trust and simplify security decision-making.

References

1. J. Bhayo, S. A. Shah, S. Hameed et al., “Towards a machine learning-based framework for DDoS attack detection in software-defined IoT (SD-IoT) networks,” *Engineering Applications of Artificial Intelligence*, vol. 123, Aug. 2023, Art. no. 106432.
2. A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, and H. Patel, “Deep learning-based classification model for botnet attack detection,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 7, pp. 3457–3466, Jul. 2022.
3. S. I. Popoola, B. Adebisi, M. Hammoudeh et al., “Hybrid deep learning for botnet attack detection in IoT networks,” *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4944–4956, Mar. 2021.
4. S. Sriram, R. Vinayakumar, M. Alazab, and K. Soman, “Network flow based IoT botnet attack detection using deep learning,” in *IEEE INFOCOM Workshops*, Jul. 2020, pp. 189–194.
5. Z. Al-Othman, M. Alkasassbeh, and S. A.-H. Baddar, “A state-of-the-art review on IoT botnet attack detection,” *arXiv preprint*, 2020.
6. M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cybersecurity intrusion detection: Approaches, datasets, and comparative study,” *Journal of Information Security and Applications*, vol. 50, Feb. 2020.
7. T. Hasan, J. Malik, I. Bibi et al., “Securing industrial IoT against botnet attacks using hybrid deep learning approach,” *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2952–2963, 2023.
8. D. T. Son, N. T. K. Tram, and P. M. Hieu, “Deep learning techniques to detect botnet,” *Journal of Science and Technology Information Security*, vol. 1, no. 15, pp. 85–91, 2022.
9. M. Gandhi and S. Srivatsa, “Detecting and preventing attacks using network intrusion detection systems,” *International Journal of Computer Science and Security*, vol. 2, no. 1, pp. 49–60, 2008.
10. J. Liu, S. Liu, and S. Zhang, “Detection of IoT botnet based on deep learning,” in *Chinese Control Conference (CCC)*, 2019, pp. 8381–8385.