ENHANCED BOTNET DETECTION IN IOT ENVIRONMENT USING STACKED DEEP LEARNING MODELS

A PROJECT REPORT - PHASE II

Submitted by

ARAVIND A

Under the guidance of

Dr. N. RAJENDRAN

In partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

MAY 2025

B.S. Abdur Rahman®
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

BONAFIDE CERTIFICATE

Certified that this project report **"ENHANCED BOTNET DETECTION IN IOT ENVIRONMENT USING STACKED DEEP LEARNING MODELS"** is the bonafide work of **ARAVIND A (231272601001)** who carried out the project work under my supervision. Certified further, that to the best of our knowledge, the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. N. RAJENDRAN
SUPERVISOR
Associate Professor
Department of Information Technology
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048

SIGNATURE

Dr. N. PRAKASH
HEAD OF THE DEPARTMENT
Professor
Department of Information Technology
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048

B.S. Abdur Rahman®
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

VIVA-VOCE EXAMINATION

The viva-voce examination of the project work titled **"ENHANCED BOTNET DETECTION IN IOT ENVIRONMENT USING STACKED DEEP LEARNING MODELS"**, submitted by **ARAVIND A (RRN: 231272601001)** is held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I sincerely express my heartfelt gratitude to **Dr. T. MURUGESAN**, Vice Chancellor and **Dr. THAJUDDIN**, Pro-Vice Chancellor, B.S. Abdur Rahman Crescent Institute of Science and Technology.

I sincerely thank, **Dr. N. RAJA HUSSAIN**, Registrar, B.S. Abdur Rahman Crescent Institute of Science and Technology, for furnishing every essential facility for doing our project.

I would also like to express my sincere gratitude to **Dr. SHARMILA SANKAR**, Dean, School of Computer, Information and Mathematical Sciences, for her support.

I thank **Dr. N. PRAKASH**, Professor and Head, Department of Information Technology for his eminent support and encouragement throughout my project.

I sincerely thank my Project Supervisor and Project Coordinator **Dr. N. RAJENDRAN,** Associate Professor, Department of Information Technology for his guidance and motivation throughout the project.

I express my sincere thanks to the Project Review Committee Members **Dr. G. KAVITHA,** Professor, **Dr. P. GNANSEKARAN,** Assistant Professor (Sel. Grade) and **Dr. A. SONYA,** Assistant Professor (Ser. Grade), Department of Information Technology for their continuous support to complete the project work successfully.

I would also like to thank my dear parents and friends who directly or indirectly helped me in the completion of the project.

ARAVIND A

ABSTRACT

The rapid expansion of the Internet of Things (IoT) has exposed these systems to growing cyber threats, particularly botnet attacks. Botnets, comprising compromised devices under a single malicious control, can disrupt IoT networks, causing security breaches, denial-of-service (DoS) attacks, and unauthorized data access. Traditional security measures and even existing machine learning- based Intrusion Detection Systems (IDS) often struggle to keep pace with the evolving nature of botnets and the large, diverse traffic in IoT environments. This project proposes an enhanced botnet detection system using a stacked deep learning model that integrates DNN and BiLSTM for botnet attack detection in IoT environments. The model custom Deep Neural Network (DNN) with a Bidirectional Long Short-Term Memory (BiLSTM) network to effectively capture temporal dependencies and complex attack patterns within network traffic data. The dataset undergoes rigorous preprocessing, including feature selection, categorical encoding, and normalization, to enhance model performance. Experimental results demonstrate the model's ability to differentiate between normal and botnet-infected traffic with high accuracy. The proposed system is implemented within a Streamlit-based web application, enabling real-time botnet attack detection through CSV file uploads. The results indicate that the stacked DNN-BiLSTM model enhances threat detection, providing a robust and scalable solution for securing IoT networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	-	Artificial Neural Network
AUC	-	Area Under the Curve
CDIS	-	Combined Disk Scheduling
CNN	-	Convolutional Neural Network
DDoS	-	Distributed Denial of Service
IOT	-	Internet of Things
LSA	-	Latent Semantic Analysis
LSTM	-	Long Short-Term Memory
RNN	-	Recurrent Neural Network ANN
ROC	-	Receiver Operating Characteristic
SDN	-	Software-Defined Networking
TNR	-	True Negative Rate
TPR	-	True Positive Rate
WSN	-	Wireless Sensor Network

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Industries have undergone a revolution thanks to the widespread use of Internet of Things (IoT) devices, which have made automation and seamless connectivity possible. These gadgets are used in a variety of fields, such as healthcare, smart homes, industrial automation, and transportation, and they help with real-time decision-making and increased efficiency. But there are also serious cyber security issues with the IoT broad adoption. Because of their inadequate security measures and computational capabilities, IoT devices are highly vulnerable to cyber threats, particularly botnet attacks. These attacks compromise network integrity, disrupt operations, and pose serious risks to sensitive data.

Botnets are groups of compromised devices under the control of malevolent actors that carry out coordinated cyberattacks, including data theft, Distributed Denial of Service (DDoS), and unauthorized access. Because of the constantly changing nature of cyberattacks and the large volume of IoT network traffic, traditional security solutions like rule-based intrusion detection systems and signature-based firewalls frequently fall short in identifying these complex threats. Furthermore, traditional methods have trouble analyzing and classifying network anomalies in real time, which results in a higher number of false positives and slower responses. Machine learning and deep learning models have gained popularity in cyber security as a way to overcome these limitations because of their ability to effectively analyze massive amounts of network traffic and identify malicious patterns.

Recent advances in artificial intelligence (AI) have led to the development of stacked deep learning models, which combine multiple neural network architectures to improve detection performance. Intrusion detection systems based on deep learning can improve IoT network security by utilizing both feature extraction and sequential pattern learning. By combining deep learning with real-time detection frameworks, botnet attacks can be identified more quickly and accurately, offering a proactive way to reduce security risks in IoT environment.

As of 2025, there are expected to be more than 25 billion IoT devices globally, which generate massive amounts of data and enable intelligent communication between devices. This interconnected environment is revolutionizing how businesses operate and how services are delivered to consumers. However, with the increasing adoption of IoT devices, there is a growing concern about the security and privacy of the networks that support them. Wireless Sensor Networks (WSN), a subset of IoT, are particularly vulnerable due to their distributed and resource-constrained nature, making them prime targets for cyberattacks, particularly botnet-based attacks.

1.2 MOTIVATION

The requirement for a botnet detection system specifically designed for IoT-WSN environments serves as the driving force. The suggested method for detecting IoT botnet attacks makes use of a Stacked Bidirectional Long Short-Term Memory (BiLSTM) and a Customized Deep Neural Network (DNN) are combined in this Deep Learning model. To ensure a diverse range of IoT network traffic data, the system starts with dataset collection from Kaggle. Pre-processing of the data is done, which includes feature, label encoding, and data cleaning. The stacked DNN-BiLSTM model is intended to use BiLSTM sequential learning capability for botnet attack detection and DNN's capacity to extract intricate patterns. Labeled network traffic data is used to train the model, and iterative learning is used to maximize performance. Following training, the model is assessed using a range of performance metrics, such as accuracy, confusion matrix, precision, recall, F1-score, and ROC curve analysis, ensuring a robust assessment of its predictive capabilities.

1.3 THE RISE OF IOT AND ITS SECURITY IMPLICATIONS

The Internet of Things (IoT) is changing industries and individual interactions with the environment through devices like smart thermostats, wearable health monitors, and industrial sensors. These interconnected devices exchange data, analyse information, and perform automated actions, delivering benefits such as improved efficiency and decision-making. However, IoT rapid growth has introduced critical security challenges.

Many IoT devices handle sensitive information, including health data and financial transactions, making security breaches potentially devastating. Despite this, IoT devices are often designed with limited resources, hindering traditional security measures like encryption and authentication. Moreover, their deployment in remote areas complicates monitoring and updates. A major threat to IoT security is botnet attacks Distributed Denial of Service (DDoS) attacks often make use of an attacker-controlled network of compromised devices.

IoT botnet exploit vulnerabilities in poorly secured devices, adding them to their network for large-scale disruptions. The infamous Mirai botnet, which caused massive internet outages in 2021, exemplifies these risks. By exploiting devices with weak passwords, it launched a significant DDoS attack, affecting services like Twitter and Netflix. This highlighted the urgent need for stronger IoT security measures to protect against such vulnerabilities.

1.4 THE CHALLENGES OF BOTNET DETECTION

Detecting botnet attacks in IoT networks presents several challenges that differ from traditional network security. These challenges include:

1. **Limited Network Traffic Data**: IoT devices often produce sparse or low-volume traffic compared to traditional computing systems.
2. **Heterogeneity of Devices**: Numerous devices with varying communication protocols, capabilities, and traffic patterns make up Internet of Things networks.
3. **Resource Constraints**: The low cost and power consumption of many IoT devices restricts their capacity to carry out intricate calculations.
4. **Data Scarcity**: Unlike traditional IT systems, where large amounts of labelled data are readily available for training detection models, IoT networks often suffer from a lack of labelled network traffic data.
5. **Real-Time Detection Requirements**: In IoT-WSNs, timely detection of botnet attacks is crucial. Delays in identifying malicious activity can allow attackers to execute further operations, such as data exfiltration or system manipulation.

6. **Adversarial Attacks**: Small, skillfully constructed perturbations are added to the input data in an effort to trick deep learning models into producing inaccurate predictions.

1.5 THE ROLE OF DEEP LEARNING

Deep learning, particularly integrating a Customized Deep Neural Network (DNN) and Bidirectional Long Short-Term Memory (BiLSTM), has shown great promise in a number of fields, including time-series analysis, image recognition, and natural language processing. The stacked DNN-BiLSTM model is designed to leverage DNN's ability to extract complex patterns and BiLSTM sequential learning capability to identify botnet attacks. Over the past few years, BiLSTM have also been applied to network traffic analysis and intrusion detection systems (IDS), with promising results.

The key advantage of using deep learning for botnet detection is its ability to automatically extract pertinent features from massive amounts of network traffic data, which is particularly important in the context of IoT-WSNs. Deep learning models can learn to differentiate between benign and malicious traffic by analyzing raw data patterns, even in the presence of noise or anomalies. This ability to automatically learn features from data makes deep learning models highly effective for botnet detection in IoT networks, where traffic patterns can vary significantly across devices and contexts.

1.6 OBJECTIVES

Create an improved botnet detection system especially for Internet of Things (IoT) environments. To improve the detection of botnet attacks in IoT environments, the system will employ a stacked deep learning model that combines a Bidirectional Long Short-Term Memory (BiLSTM) network with a Customized Deep Neural Network (DNN). Apply categorical encoding and normalization to optimize network traffic data for improved classification accuracy. Train the model on diverse IoT network traffic datasets to improve its generalization and ability to distinguish between normal and malicious activities.

Design a Streamlit-based web application that allows real-time IoT botnet attack detection through CSV file uploads, facilitating ease of use and accessibility.

1. **Develop a Stacked Deep Learning Model**: Implement a stacked deep learning model integrating a Customized Deep Neural Network and Bidirectional Long Short-Term Memory to improve feature extraction and classification accuracy for identifying botnet activity in IoT-WSNs.
2. **Utilize Established Datasets**: Train and evaluate the model using the datasets to ensure robust performance in detecting botnet attacks, while adapting it for real-world IoT traffic data.
3. **Achieve Real-Time Detection**: Optimize the system to minimize computational overhead and latency, ensuring it is capable of detecting botnet attacks in real-time on resource-constrained IoT devices.
4. **Ensure Robustness Against Adversarial Attacks**: Develop strategies to protect the model against adversarial attacks that could compromise detection accuracy, ensuring the system remains secure and reliable under various attack scenarios.
5. **Optimize Scalability and Efficiency**: Balance model complexity and system performance, ensuring that the botnet detection system can scale efficiently to handle large IoT networks without compromising accuracy.

1.7 ORGANIZATION OF THESIS

The thesis is set up to offer a thorough analysis of the suggested work. in privacy preservation, data integrity, and confidentiality in healthcare cloud computing environments. The following outlines the organization of thesis:

Chapter 1 The research topic is introduced, which includes background data and highlighting the challenges related to privacy, data integrity, and confidentiality. It also presents the motivation behind the proposed work and outlines the contributions made by the researcher for the study.

Chapter 2 provides a comprehensive review of relevant literature is conducted, exploring existing approaches, techniques, and methodologies related to privacy preservation, data integrity, analyzing existing research findings and

identifying gaps or limitations in current approaches.

Chapter 3 discusses the limitations of current botnet detection systems, focusing on the lack of labelled data, vulnerability to adversarial attacks, and scalability issues. It evaluates the performance of an existing model (ACLR) and highlights its accuracy and robustness. Proposed System: Introduces a stacked deep learning model integrate Customized DNN and BiLSTM ResNet50 to improve detection accuracy, scalability, and real-time processing in IoT-WSN environments. The system leverages advanced feature extraction and includes defenses against adversarial attacks.

Chapter 4 describes the integration of Customized DNN and BiLSTM models for feature extraction and classification, adapted for network traffic data. Google Colab is used for implementation, including model fine-tuning and system evaluation. Key steps includes; data pre-processing, feature extraction, training and deployment of the deep learning model for real-time detection.

Chapter 5 assesses the system's effectiveness using datasets and simulated traffic, with key outcomes like high detection accuracy and real-time capabilities. Outputs: Presents visualizations and tables from simulations, showcasing the system's ability to detect botnet activities and generate actionable insights.

Chapter 6 highlights the successful development of a robust botnet detection system leveraging Customized DNN and BiLSTM. The system addresses IoT-WSN-specific challenges, offering real-time, scalable, and efficient detection, Significance. Emphasizes the model's adaptability to real-world IoT traffic and its defenses against adversarial attacks, contributing to enhanced IoT security.

CHAPTER 2
LITERATURE REVIEW

2.1 SURVEY OF EXISTING SYSTEMS

Computer network attacks have the ability to read, corrupt, and steal data, which has a catastrophic effect on the overall performance of the system. Pre-intrusion activities like port scanning and IP spoofing come before attacks. Attack detection is aided by monitoring data from source and destination IP addresses, ports, protocol specifics, header specifics, etc. Attacks can be categorized as either passive or active based on their nature. The passive attack, which can be network-based or system-based, entails the hacker surreptitiously monitoring the network to obtain personal data. It could be challenging to monitor passive attacks. By exploiting security holes, impersonating a trustworthy system, or stealing credentials, active attackers get around all security measures and access networks.

Machine Learning Approaches for Botnet Attack Detection

The ability of recurrent neural networks (RNNs) to accurately identify network traffic samples from highly unbalanced classes has been studied. SRNN performed better than RNN when tested on the Bot-IoT dataset. RNN is used to train feature representations of highly skewed network traffic data for discriminative categorization. GM, MCC, F1, AUC, precision, and recall are all enhanced by the SRNN model. Spam emails are beginning to have a significant impact on networks and user productivity. For the selected datasets, estimates of the F score and average validation accuracy were calculated, as well as the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and true negative rate (TNR). The results show that the deep RNN technique can achieve up to 98.65% accuracy and detection rate.

Traffic separation prevents major movement, and DDoS detection systems differentiate between streams of regular and abnormal activity. Security flaws and centralized control make SDNs susceptible. It has been investigated whether network traffic samples from highly unbalanced classes can be reliably identified

by recurrent neural networks (RNNs). When tested on the Bot-IoT dataset, SRNN outperformed RNN. For discriminative categorization, highly skewed network traffic data feature representations are trained using RNN. The SRNN model improves GM, MCC, F1, AUC, precision, and recall. Networks and user productivity are starting to be significantly impacted by spam emails. True positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and true negative rate (TNR) were computed for the chosen datasets, along with estimates of the F score and average validation accuracy.

The author highlights scalability, energy and security management, load balancing, and fault tolerance in SDN management solutions. Intelligent decision-making in uncertain circumstances requires more than just AI-based techniques. AI, IoT, and block chain can enhance security, privacy, and openness. The rapidly growing Internet of Things (IoT) networks provide security concerns that can be addressed by the Smart Data Network (SDN) strategy.

A layered machine learning model called ACLR was introduced by M. Ali et al. [1] to identify botnet assaults in Internet of Things networks. Long short-term memory (LSTM), recurrent neural networks (RNN), convolutional neural networks (CNN), and artificial neural networks (ANN) are all combined into a stacked ensemble architecture in the ACLR model. By capturing intricate patterns and traits in botnet network data, this layered technique greatly increases the accuracy of detection. Nine distinct attack types are included in the UNSW-NB15 dataset, which the authors use to assess ACLR's performance. The model's K-fold cross-validation accuracy score of 0.9749 indicates that it excels in generalizability and achieves a remarkable test accuracy of 96.98%. Furthermore, with precision-recall area under the curve (PR-AUC) and receiver operating characteristic (ROC-AUC) scores of 0.9934 and 0.9950, respectively, the ACLR model achieves outstanding results. These results highlight the model’s superior performance compared to other state-of-the-art approaches for botnet detection. The research demonstrates that the ACLR model can effectively address the challenges posed by the evolving nature of botnet attacks, offering a robust solution for enhancing cybersecurity in IoT networks.

A deep reinforcement learning (RL) model called MalBoT-DRL was proposed by M. Al-Fawa'reh et al. [2] with the goal of detecting botnets in IoT networks at every stage of their lifespan. The model overcomes the drawbacks of conventional machine learning methods, which have trouble with generalizability and model drift when faced with changing virus patterns. MalBoT-DRL is very effective in real-world IoT situations because it dynamically adjusts to novel or unforeseen threats by fusing damped incremental statistics with an attention reward mechanism. By gradually improving the model's capacity to discriminate between benign and malevolent network activity, this method improves detection performance. To confirm the efficacy of the model, the authors perform trace-driven experiments on two well-known datasets, MedBIoT and N-BaIoT. In the early and late detection phases, they achieve impressive average detection rates of 99.80% and 99.40%, respectively. The paper shows how deep reinforcement learning can be used to overcome model drift and providing a more resilient IDS for dynamic IoT environments. MalBoT-DRL stands out for its ability to adapt to new threats, making it a significant advancement in the detection of Botnet and other malware in IoT networks. The significance of generalizability in intrusion detection systems for changing cyber threats in the Internet of Things is emphasized in the research.

F. Taher et al. [3] proposed a stacked feature selection algorithm (FGOA-kNN) based on a combination of techniques to detect botnets in Industrial IoT (IIoT) scenarios and wrapper selection methods. This approach leverages the Grasshopper Optimization Algorithm (GOA) to rank features and minimize the most relevant ones. Additionally, the Harris Hawks Optimization (HHO) algorithm is used to tune hyper parameters for better botnet detection. The HHO algorithm is enhanced with several improvements to better explore the search space and avoid local minima, utilizing a chaotic map function for initialization and an elite operator to improve exploitation. The model combines unsupervised clustering techniques with supervised learning approaches to improve the robustness and accuracy of detection. High accuracy detection of known and new botnet attacks is made possible by this stacking method. The proposed model is validated using the N-BaIoT dataset, showing significant improvements over existing methods in terms of detecting multiclass botnet attacks. The findings show that in IIoT contexts, the suggested stacking feature selection and optimization methods, in conjunction with

the mix of supervised and unsupervised learning approaches, offer better performance and dependability. This work provides a strong method for botnet attack detection, which helps to improve security in IIoT systems.

M. S. Alshehri et al. [4] presented SkipGateNet, a ground-breaking deep learning architecture, to identify Mirai and Bashlite botnet attacks in resource-constrained IoT and fog computing environments. In order to optimize the detection process, the model combines layers of Long Short-Term Memory (LSTM) and 1D-Convolutional Neural Networks (CNN) with learnable skip connections. By adding gating techniques to these skip connections, the model is able to give important features priority. while eliminating those that are unnecessary, hence enhancing performance in environments with limited resources. SkipGateNet operates with a high detection accuracy of 99.91% and a small size of 2596.87 KB, SkipGateNet functions incredibly well and is appropriate for real-time deployment in Internet of Things networks. The system has an inference time of under 8.0 milliseconds when evaluated using the N-BaIoT dataset. Its performance is validated using metrics such as precision, recall, accuracy, and F1 score in addition to statistical measures like Cohen's Kappa Coefficient and Matthews Correlation Coefficient. By tackling the difficulties of botnet identification, SkipGateNet is a viable approach for enhancing security in IoT contexts. The results show that it outperforms other models, including different CNN and CNN-LSTM designs.

R. Kalakoti et al. [5] addressed the challenge of detecting botnet attacks in IoT networks by focusing on feature selection for machine learning models. The authors propose minimizing the feature sets used in machine learning tasks by creating six distinct classification issues according to different phases of the botnet life cycle. They accomplish this by selecting the best feature set for every classification task using filter and wrapper approaches. The experimental findings show that a notably smaller number of characteristics can nonetheless yield good detection rates, indicating the efficiency of feature selection techniques. The study finds that some wrapper methods guarantee optimal feature sets regardless of the problem formulation, while filtering techniques don't always produce the same outcome. Additionally, the study emphasizes the significance of several feature types at different phases of the botnet life cycle. While host-based features are

more successful in detecting assaults coming from compromised bots, channel-based features are shown to be essential for detection in the post-attack, communication, and control stages. By decreasing the computational complexity and enhancing performance with fewer features, this method aids in the creation of more effective machine learning models for botnet detection in IoT contexts.

A novel pipeline for IoT botnet detection was presented by D. Arnold et al. [6] that combines Convolutional Neural Networks (CNNs) and network traffic visualization. By turning unprocessed network traffic data into visual representations, this technique makes use of CNNs' capabilities in picture categorization, which are then processed by the CNN for botnet detection. The approach aims to overcome the latency and resource consumption challenges typically associated with traditional Intrusion Detection Systems (IDS) that process network traffic at the center of the network. By moving the detection process to the network's edge, the proposed system improves response times and reduces strain on network resources. The pipeline achieves remarkable results when tested on the N-BaIoT and IoT-23 datasets, with detection rates of 100% and 99.78%, respectively. Furthermore, the CNN-based approach offers significant improvements in throughput, processing 2.4 times more packets per second compared to traditional models. Additionally, the proposed system is more efficient, with a 21.4% reduction in model size compared to a similar Deep Neural Network (DNN) model. The suggested pipeline appears to be a viable option for real-time botnet detection in IoT systems, particularly those with constrained resources, based on these performance enhancements and high detection accuracy.

In order to achieve effective warning correlation, M. Lefoane et al. [7] employ a creative two-phase method for IoT botnet detection that combines Latent Semantic Analysis (LSA) and graph theory. In order to differentiate malicious traffic from legitimate traffic, the system's initial phase uses feature selection and classification techniques to identify botnet traffic. To find unique botnet attack campaigns, the second stage uses alert correlation. This phase utilizes LSA, an unsupervised learning technique, and graph theory to analyze the structure and relationships within the network traffic, improving the ability to identify botnet attacks. Using actual IoT traffic statistics, the authors test the system and obtain

remarkable results, with a False Positive Rate (FPR) of 0% and a True Positive Rate (TPR) of over 99%. The aforementioned outcomes illustrate how well the technology detects IoT botnet assaults while reducing false alarms. The suggested method is a useful tool for IoT intrusion detection systems since it provides a fresh method of correlating alarms from various sources. This study demonstrates how semantic analysis and graph theory can be combined to handle the particular difficulties presented by IoT botnet attacks.

S. A. Abdul Kareem et al. [8] conducted a comprehensive survey on network intrusion detection (NID) systems, focusing on both IoT and non-IoT environments. With the rapid expansion of IoT, security vulnerabilities in these networks are becoming more evident, especially due to the exploitation of connected devices in botnet attacks. The survey emphasizes how important it is to have efficient intrusion detection systems that are adapted to the unique features of IoT networks. The authors analyze IoT-specific NID techniques, evaluating various datasets, tools, and machine learning classifiers employed in IoT security. The survey covers research from 2018 to 2024, providing a detailed review of the methods used to detect and mitigate security breaches in IoT systems. A significant portion of the review is dedicated to machine learning classifiers, particularly supervised learning methods, which have shown high success rates in detecting security threats in both IoT and traditional networks. The study also identifies publicly available IoT datasets that can be used for NID experiments, facilitating academic and industrial research. In addition, the paper offers insights into the ongoing problems and opportunities in the field of IoT intrusion detection by discussing existing trends and possible future developments in IoT security.

A new online method called the Compromised Device Identification method (CDIS) was presented by E. Gelenbe and M. Nakıp [9] to identify compromised IoT devices during botnet attacks. This system uses traffic metrics, which are easily extracted from network traffic, to train itself during normal operations using an Auto-Associative Dense Random Neural Network (AADRNN). Since the system learns from traffic that is thought to be benign, it does not require the acquisition of attack traffic data beforehand. The technique uses online, auto-associative learning to save training time and continuously enhance performance. Based on publicly

accessible Mirai Botnet attack data, the experimental evaluation shows that, in spite of its short training and execution times, CDIS achieves a balanced accuracy of 97%. The findings demonstrate that when it comes to identifying infected devices, AADRNN with sequential learning performs better than six other cutting-edge machine learning models. Because of this, CDIS is a viable strategy for stopping botnet attacks from spreading throughout IoT networks that have several devices and IP addresses. CDIS improves the capacity to defend IoT systems against botnet infiltration and lessens the possible harm brought on by such assaults by offering real-time device identification of compromised devices. This study shows how the AADRNN technique and sequential learning may guarantee reliable botnet detection in Internet of Things settings.

A layered deep learning technique was presented by F. Sattari et al. [10] to identify bottleneck attacks in IoT systems, especially in fog computing settings. As a cloud computing extension, fog computing lowers latency and boosts efficiency by enabling IoT devices to process data locally. However, the presence of resource-constrained devices in IoT networks makes them susceptible to cyber-attacks like Botnet, Denial of Service (DoS), and Distributed Denial of Service (DDoS). The authors propose an innovative solution that leverages software-defined networking (SDN) to identify and stop attacks instantly. With a remarkable 99.98% detection accuracy, the stacked deep learning model combines several methods to detect complex botnet attacks. This solution also offers excellent speed, with a processing time of just 0.022 milliseconds, ensuring minimal impact on network performance. The effectiveness of the method is demonstrated through extensive testing on the latest IoT datasets, showing that the proposed system outperforms existing models.

The results of the study highlight how crucial it is to combine deep learning and SDN in order to identify and stop criminal activity in IoT networks. The protection of fog computing-based IoT environments against new cybersecurity risks, especially botnet-driven attacks, is greatly advanced by our work.

2.2 SUMMARY OF EXISTING SYSTEMS

The majority of today's IoT security systems rely on antiquated techniques like rule-based and signature-based intrusion detection, which are useless against unidentified attackers. Machine learning approaches have been introduced but struggle with the need for large labelled datasets and limited generalization. Deep learning methods, while more accurate, are computationally intensive and often compromise data privacy due to centralized processing. Federated learning helps preserve privacy by training models at the edge but faces challenges with non-uniform data and device variability. Cloud-based security systems offer scalability but introduce latency and privacy concerns. Overall, these systems lack the adaptability, efficiency, and real-time capabilities needed for robust IoT security.

CHAPTER 3
PROBLEM DEFINITION

3.1 EXISTING SYSTEM

This study proposes a botnet identification system based on the stacking of Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNN) (ACLR). The experiments make use of both the individual models and the proposed ACLR model for performance comparison. The UNSW-NB15 dataset, used for botnet attacks, contains nine different attack types: "Normal," "Generic," "Exploits," "Fuzzers," "DoS," "Reconnaissance," "Analysis," "Backdoor," "Shell code," and "Worms." Experimental results show that the proposed ACLR model achieves a testing accuracy of 0.9698, effectively capturing the intricate patterns and characteristics of botnet attacks. The proposed ACLR model's k values (3, 5, 7, and 10) for a K-fold cross-validation accuracy score of 0.9749 show that k = 5 illustrates the model's generalizability and resilience. Additionally, the proposed model detects botnets with a high receiver operating characteristic area under the curve (ROC-AUC) of 0.9934 and a precision-recall area under the curve (PR-AUC) of 0.9950. A comparison with existing state-of-the-art models further demonstrates the superior performance of the proposed method. The results of this study can help combat emerging threats and enhance cyber security procedures.

3.1.2 CHALLENGES IN EXISTING SYSTEMS

Despite the various approaches to botnet detection, existing systems still face several significant challenges, especially in the context of IoT networks:

1. **Lack of Labelled Data**: One of the biggest challenges is the scarcity of labelled IoT traffic data in order to train models for machine learning. The absence of comprehensive datasets that accurately represent real-world IoT traffic makes it difficult to develop robust models that can detect both known and novel attacks.

2. **Adversarial Attacks**: As botnet evolve, attackers increasingly employ adversarial techniques to evade detection. These attacks manipulate the input data to mislead detection systems, making it harder for security mechanisms to identify malicious behavior.
3. **Scalability**: Thousands or even millions of devices can make up an IoT network, making it challenging for detection systems to scale effectively without compromising performance. A system that works well in small-scale IoT environments may struggle to maintain efficiency and accuracy in large, distributed networks.

To address these challenges, there is a need for innovative detection systems that can overcome the limitations of existing approaches while being tailored to the unique characteristics of IoT environments.

3.1.3 EXISTING SYSTEM DESIGN

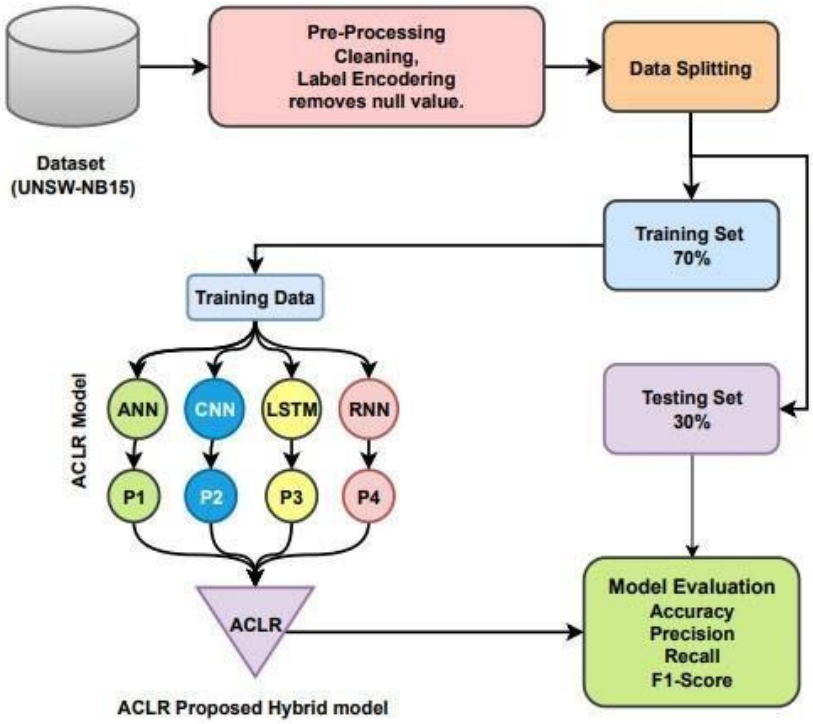

Fig. 3.1 Existing System Design

3.2 PROPOSED SYSTEM

The proposed method for detecting IoT botnet attacks makes use of a Stacked Deep Learning Model that combines Bidirectional Long Short-Term Memory (BiLSTM) with a Customized Deep Neural Network (DNN). The dataset collection from Kaggle, ensuring a diverse range of IoT network traffic data. Data pre-processing is performed, including data cleaning to handle missing or irrelevant attributes, label encoding for categorical variables such as protocol and flag fields, and feature scaling using Standard Scaler to normalize numerical values. To guarantee a fair assessment, the dataset is then divided into training (80%) and testing (20%) subsets.

The stacked DNN-BiLSTM model is designed to leverage DNN's capacity to identify botnet attacks using BiLSTM sequential learning and the ability to extract intricate patterns. The model training, the model is assessed using a range of performance metrics, such as ROC curve analysis, F1-score, confusion matrix, accuracy, precision, and recall, ensuring a robust assessment of its predictive capabilities. Finally, the trained model is integrated into a Streamlit web framework, allowing users to upload IoT network traffic data in CSV format and obtain real-time predictions on whether the traffic is normal or an attack. The botnet detection system's usability and accessibility are improved by its intuitive interface.

3.2.1 PROBLEM IDENTIFICATION

The rapid expansion of IoT devices has significantly increased network connectivity, making them a leading target for cyberattacks, particularly botnet attacks. Botnet can compromise IoT networks by hijacking devices and executing extensive malevolent actions like data breaches, Distributed Denial of Service (DDoS) attacks, and unapproved remote control. Conventional intrusion detection systems have trouble effectively identify these sophisticated attacks due to the dynamic and evolving nature of network traffic patterns.

Despite the critical need for enhanced security in IoT Wireless Sensor Networks, existing detection systems fall short due to inadequate data, resource limitations, and evolving attack methodologies. This project aims to address these

shortcomings by developing a sophisticated botnet detection system that leverages advanced deep learning techniques.

3.2.2 DATASET COLLECTION FOR PROPOSED SYSTEM

In this work, the dataset was sourced from Kaggle, specifically the "Bot-IoT" dataset available at https://www.kaggle.com/datasets/vigneshvenkateswaran/bot-iot. The Bot-IoT dataset is widely recognized for its extensive and realistic representation of IoT network traffic, encompassing a substantial volume of labeled data corresponding to both benign activities and various types of botnet attacks, including Distributed Denial of Service (DDoS), Denial of Service (DoS), data theft, and reconnaissance attacks. In this work, particular emphasis was placed on reconnaissance attacks, which involve scanning and probing techniques commonly employed by adversaries to gather critical information about target systems. This targeted selection ensures that the developed model is specifically trained and evaluated on patterns associated with reconnaissance behavior, thereby enhancing its specialization and detection accuracy. The diversity and authenticity of the Bot-IoT dataset make it an ideal foundation for developing and validating advanced machine learning and deep learning models aimed at fortifying IoT security.

3.2.3 ADVANTAGES OF PROPOSED SYSTEM

1. **High Detection Accuracy**: By BiLSTM and Customized, the system benefits from superior feature extraction and classification performance, significantly improving the accuracy of botnet attack detection compared to traditional methods.

2. **Adversarial Attack Defense**: The system includes safeguards against adversarial attacks, ensuring that its accuracy remains intact even in the presence of malicious efforts to bypass detection.

3. **Reduced Computational Overhead**: By utilizing a stacked deep learning model, the system maintains a balance between performance and resource usage, making it suitable for deployment on IoT devices with limited computational capabilities.

3.3. ARCHITECTURE DESIGN

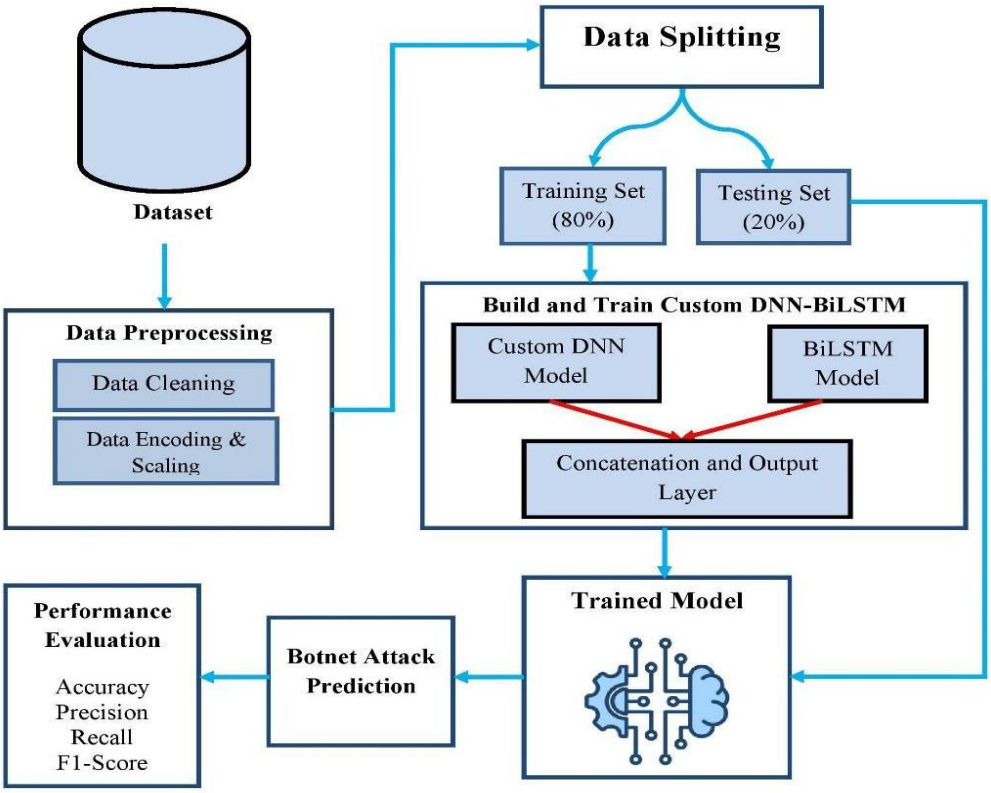

Fig. 3.2 Architecture Design

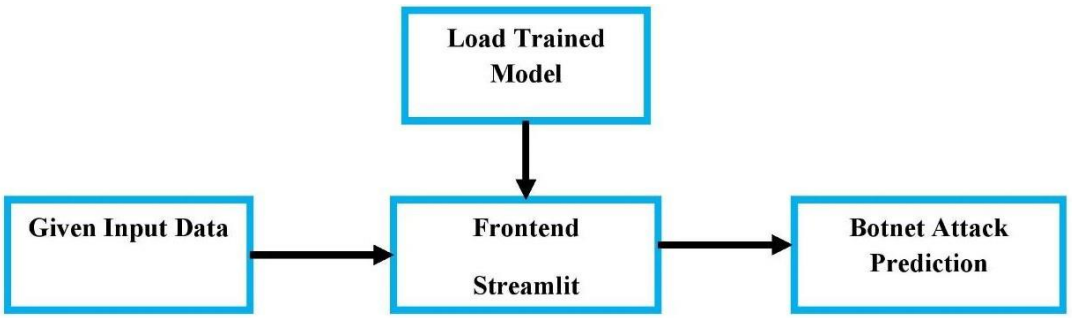

Fig. 3.3 Web Interface Design

3.4 MODULES

3.4.1 Data Collection and Pre-processing

The dataset was collected from Kaggle and contains a significant amount of IoT network traffic data with labeled records of both normal and attack instances. The dataset includes source/destination addresses, protocol types, packet sizes, and timestamps, among other features that help identify suspicious network

activity. This dataset is used to train and evaluate the botnet detection model. To improve its quality and usability, the data is extensively pre-processed before being fed into the model. Among the pre-processing actions are:

Repetitive, missing, or superfluous columns like IP addresses, timestamps, and packet sequence IDs that don't aid in attack detection are removed as part of the data cleaning process. This ensures the model correctly interprets non-numeric features:

- Label Encoding: Using a label encoder that has already been trained, categorical attributes such as protocol types and flag states are converted into numerical values.
- Feature Scaling: Standardization of numerical attributes using Standard Scaler, which scales data to have zero mean and unit variance. This helps optimize the learning process by ensuring balanced weight updates.

3.4.2 Model Building

A stacked deep learning model is designed, combining DNN and BiLSTM to leverage both feature-based and sequential learning. Model Architecture: The model consists of multiple fully connected layers, LSTM layers, and a final sigmoid activation function to classify traffic as either normal or botnet attack.

- Deep Neural Network (DNN): Extracts meaningful high-level representations from network traffic features.
- Bidirectional LSTM (BiLSTM): Captures temporal dependencies in network traffic data by analyzing packet sequences from both forward and backward directions.

3.4.3 Model Training

The stacked DNN-BiLSTM model is trained using the preprocessed dataset with the following training parameters:

- Epochs: 25 (to ensure optimal convergence and prevent overfitting)

- Batch Size: 32 (to balance computational efficiency and model performance)
- ADAM (Adaptive Moment Estimation) was selected as the optimizer due to its capacity to dynamically modify learning rates., ensuring efficient and stable training. During training, the model learns network traffic patterns by minimizing loss and adjusting weights to improve classification accuracy.

3.4.4 Detection

The Detection Module is responsible for continuously monitoring network traffic in real-time and detecting potential botnet activity using the trained stacked deep learning model. Following model training in the Model Training Module, the Detection Module, where it processes incoming traffic data and classifies it as either normal or malicious. The detection process involves evaluating the features extracted from the real-time traffic against the knowledge gained during the model's training phase. Once traffic is collected and processed, the system uses the trained model to identify patterns that match those indicative of botnet activity, such as unusual traffic spikes, repeated failed connection attempts, or irregular packet behaviors.

3.4.5 Reporting

The Reporting Module is responsible for generating detailed reports on detected threats and system performance, providing administrators with insights into the botnet detection system's effectiveness. The module aggregates information from the Detection Modules, compiling data on detected botnet activities, false positives, network performance, and overall system accuracy. The reports generated by this module can include a variety of metrics, such as the number of botnet attacks detected, the types of attacks, the devices affected, and the time taken for detection.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 IMPLEMENTATION OF THE PROPOSED SYSTEM

The stacked Customized Deep Neural Network (DNN) and a Bidirectional Long Short-Term Memory (BiLSTM) network to detect botnet attacks in IoT network traffic.

4.1.1 DEEP NEURAL NETWORK COMPONENT

The DNN component extracts high-level feature representations using a series of fully connected layers with neurons in the sequence of 512 → 256 → 128 → 64 → 32. Each dense layer is followed by a dropout layer (rate: 0.3) to reduce overfitting and enhance generalization. This Fig. 4.1 structure ensures that the model learns complex feature representations while maintaining robustness.

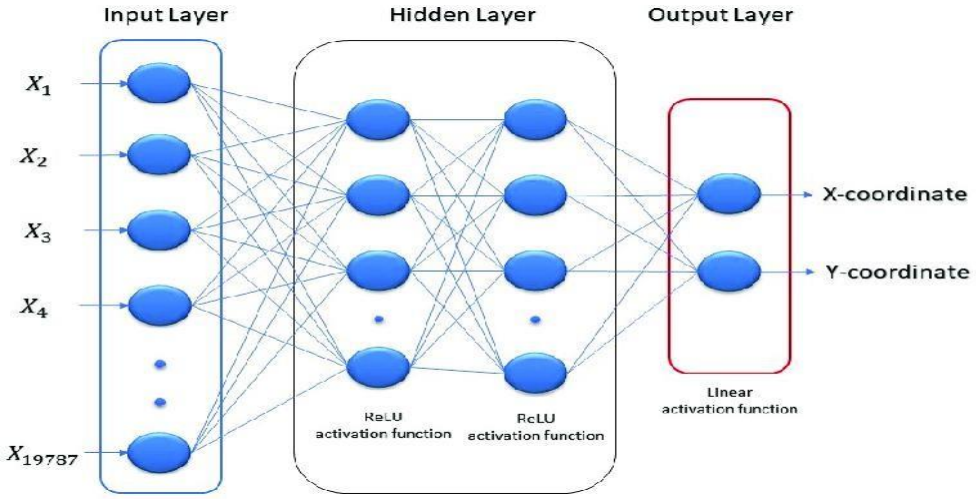

Fig. 4.1 Deep Neural Network Data Processing

4.1.2 BiLSTM COMPONENT

To capture the temporal dependencies within IoT traffic data, a Bidirectional LSTM (BiLSTM) network is employed. Fig. 4.2 like standard LSTM, the BiLSTM processes data both forward and backward, allowing it to learn relationships between past and future traffic patterns. The output from the BiLSTM layer is then

passed through two additional dense layers (each with 32 neurons) to refine extracted features further. A particular kind of recurrent neural network (RNN) called a bi-LSTM can handle sequential data in both forward and backward directions. The model is able to understand the past and future context of the input sequence by fusing the power of LSTM with bidirectional processing. Let's examine Bi-LSTM elements and operation to better understand and functionality:

- Long Short-Term Memory (LSTM): An RNN type that makes use of gating mechanisms and memory cells to retain or forget information, effectively capturing long-term dependencies in sequences.
- Bidirectional Processing: Bi-LSTM processes input in both forward and backward directions using two separate LSTM layers, each maintaining its own memories and secret states.
- Forward Pass: Using the current input and prior state as a guide, the forward LSTM updates hidden states and memory cells as it reads the input from beginning to end.
- Backward Pass: By processing the input in reverse, the backward LSTM updates its states in a comparable manner.
- Combining States: Hidden states from both directions are combined (e.g., concatenated) at each time step for richer context.

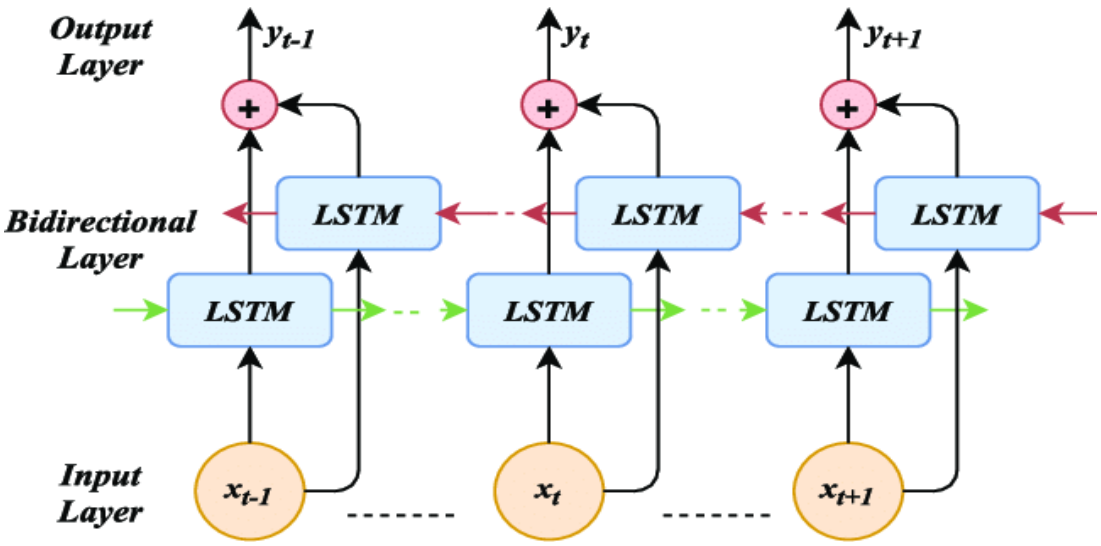

Fig. 4.2 BiLSTM Sequential Data Processing

4.1.3 FINAL PREDICTION

The outputs from both the DNN and BiLSTM components are concatenated, forming a unified model. By combining sequential dependencies (BiLSTM) and non-sequential high-level features (DNN), the model is able to produce predictions that are more accurate. The final classification is performed by a dense output layer with a single neuron, which determines whether the network traffic is normal or a botnet attack.

4.2 DNN-BiLSTM MATHEMATICAL ALGORITHM EXPLANATION

STEP 1: Input Representation

Let the input feature vector for each sample be:

$$x \in R^n$$

After reshaping for time series (with 1-time step for each sample):

$$X \in R^{1 \times n}$$

STEP 2: Modal Build

- **Deep Neural Network (DNN) Path**

This path applies multiple dense layers with ReLU activations and dropout:

Let: $h0=X$

Then each layer is defined recursively as:

$$h_1 = \text{ReLU}(W_1 h_o + b_1),\ h_1 \text{ Dropout } (h_1, p = 0.3)$$

$$h_2 = \text{ReLU}(W_2 h_1 + b_2),\ h_2 \text{ Dropout } (h_2, p = 0.3)$$

……

$$h_5 = \text{ReLU}\ (W_5 h_4 + b_5) \in R^{32}$$

- **Bidirectional LSTM Path**

For the same input X, Bi-LSTM computes forward and backward hidden states:

Let:

$h = \text{LSTM}_{Fwd}(X)$

$h = \text{LSTM}_{Bwd}(X)$

Then concatenate:

$h_{bilstm} = [h; h] \in R^{2d}$

This is followed by:

$Z_1 = \text{ReLU}\ (W_6 h_{bilstm} + b_6)$

$Z_2 = ReLU\ (W_7 Z_1 + b_7) \in R^{32}$

STEP 3: concatenated Layer

The DNN and Bi-LSTM are concatenated:

$f = [h_5; Z_2] \in R^{64}$

Final output layer (binary classification with sigmoid): $\hat{y}$

$= \sigma\ (W^T f + b) \in [0, 1]$

STEP 4: Representation of Accuracy

Let:

N: total number of samples

yi ∈ {0, 1}: true label for the it i^{th} sample

yi ∈ {0, 1}: predicted label for the i^{th} sample (after applying threshold, typically 0.5)

Then, the accuracy is defined as:

Accuracy = $1/N \sum_{i=1}^{N} 1(yi = yi)$

STEP 5: Loss Function

Binary cross-entropy is used:

$L=-(Y \log(y\hat{}) +(1-y) \log(1-y\hat{}))$

4.3 IMPLEMENTATION TOOLS

The stacked model integrates a Customized Deep Neural Network (DNN) with a Bidirectional Long Short-Term Memory (BiLSTM) network to accurately detect botnet attacks in IoT network traffic. Implemented using Google Colab for model training and Streamlit for a user-friendly web-based interface, this system enables real-time monitoring and visualization of IoT security threats.

4.3.1 GOOGLE COLAB

The free cloud-based application Google Colab, which stands for Colaboratory, enables users to write and execute Python code in an interactive, team-based environment. Because it provides users with powerful computational capabilities, it is especially well-liked for activities involving deep learning, data analysis, and machine learning.

Getting Started with Google Colab:

1. Creating a new notebook:

    - Go to Google Colab. You can create a new notebook or open an existing one from Google Drive, GitHub, or upload a local file.

2. Running code:

    - You can write Python code in the cells and run them by pressing Shift + Enter or by clicking the Play button.

3. Installing libraries:

    - You can install any additional libraries you need by running! pip install <library-name> in a code cell.

4.3.2 STREAMLIT

With the help of the open-source Python framework Streamlit, developers and data scientists can easily create and implement interactive online apps. It is particularly popular for creating data-driven applications, visualizations, and machine learning model interfaces without needing extensive knowledge of web development technologies like HTML, CSS, or JavaScript.

Key steps of Streamlit:

1. Imports necessary libraries for UI, data processing, model loading, and predictions.
2. Loads trained model (botnet_model.h5) and preprocessing tools (label_encoders, scaler) using joblib.
3. Displays app title with Customized styling using HTML inside st.markdown().
4. Accepts CSV file upload from the user for prediction input.
5. Reads uploaded data and displays a preview of the first few rows.
6. Preprocesses the data:

- Drops unused columns.
- Handles missing values.
- Encodes categorical features (proto, flgs).
- Scales numerical features.
- Reshapes input for BiLSTM model.

7. Predicts attack type using the deep learning model when the user clicks the "Predict" button.
8. Displays prediction result (either "Normal Traffic" or "Botnet Attack") on the UI.
9. Displays project description in the sidebar with a brief explanation.
10. Includes error handling to display user-friendly messages in case of issues.

CHAPTER 5
RESULTS AND DISCUSSIONS

The experimental results demonstrate the ability of the suggested stacked DNN-BiLSTM model to identify botnet assaults in Internet of Things network traffic. This result in a 1.93% improvement in detection accuracy compared to baseline approaches, achieving 98.91% accuracy and 0.9894% score of ROC AUC, the model demonstrated its great dependability in differentiating between malicious and legitimate traffic. Furthermore, the Fig. 5.1 model's high capacity to differentiate between legitimate and botnet attack traffic is demonstrated by its ROC AUC score of 0.9894, which supports its potential for practical use in IoT security frameworks.

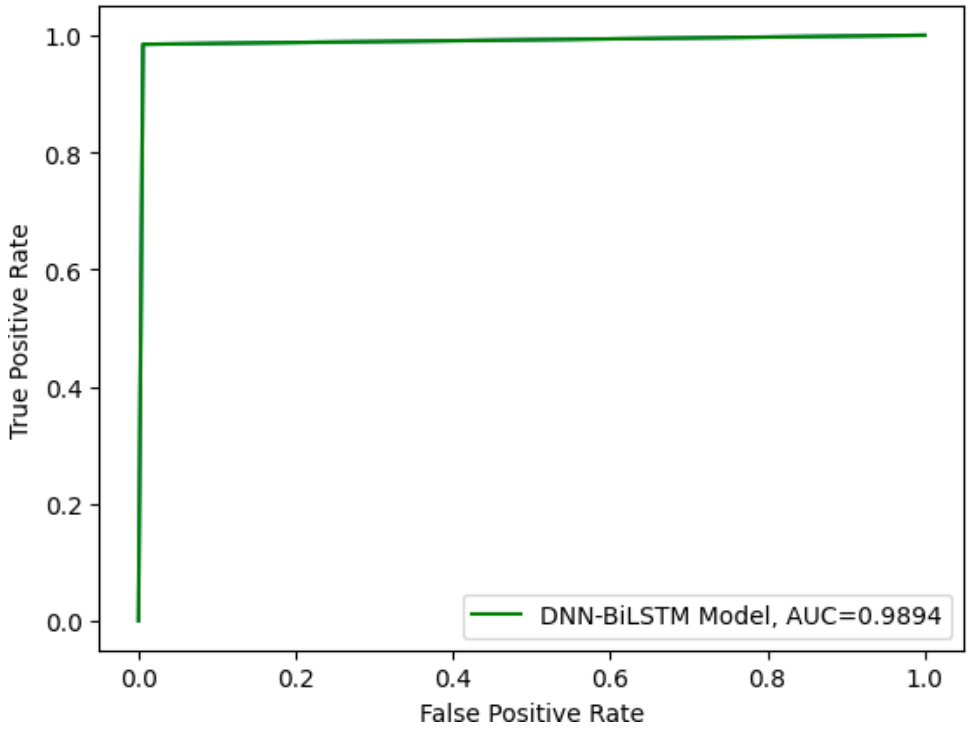

Fig. 5.1 ROC Curve

Fig. 5.2 shows the precision (0.9828 for normal and 0.9948 for attack traffic), recall (0.9942 for normal and 0.9845 for attack traffic), and F1-score (0.9885 for normal and 0.9896 for attack traffic) of the model, which ensures that both false positives and false negatives are minimized. The weighted average and macro F1-scores of 0.9891 further support the model's robustness.

Accuracy on test set: 98.91%

	precision	recall	f1-score	support
0	0.9828	0.9942	0.9885	345
1	0.9948	0.9845	0.9896	388
accuracy			0.9891	733
macro avg	0.9888	0.9894	0.9891	733
weighted avg	0.9892	0.9891	0.9891	733

Fig. 5.2 Performance Metrics

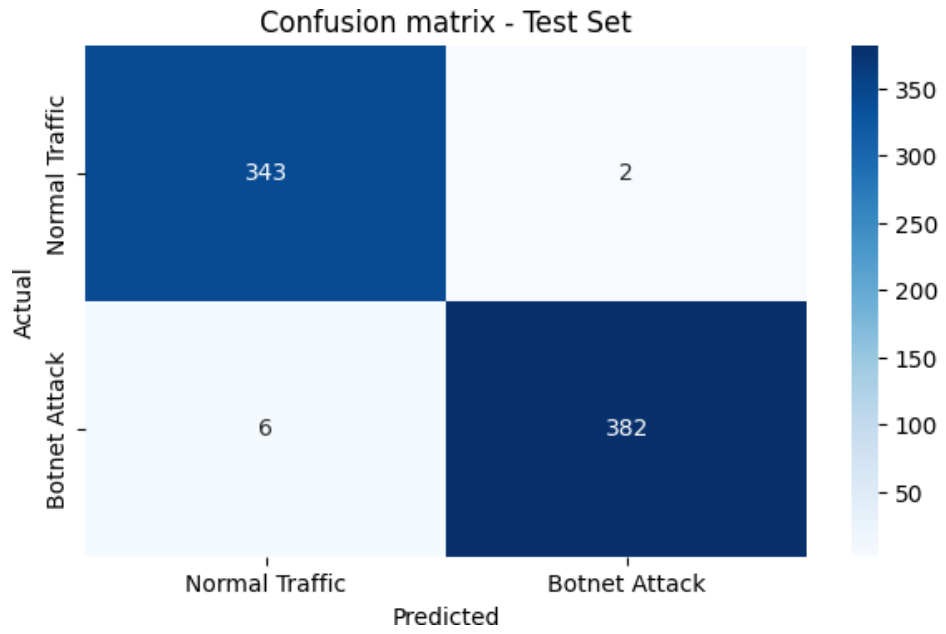

Fig. 5.3 Confusion Matrix

CHAPTER 6
CONCLUSION AND FUTURE WORK

Using a Stacked Deep Learning model that combines a Deep Neural Network (DNN) and a Customized Bidirectional Long Short-Term Memory (BiLSTM) network, we created a reliable and effective IoT botnet attack detection system in this work. The suggested model successfully captures both high-level feature representations and sequential dependencies within IoT network traffic, improving the detection accuracy of botnet attacks.

Extensive experimentation and evaluation demonstrated the effectiveness of our approach. The model was trained and tested using a well-curate dataset sourced from Kaggle, undergoing rigorous pre-processing steps, including label encoding, data normalization, and feature selection. The suggested Stacked DNN-BiLSTM model result in a 1.93% improvement in detection accuracy compared to baseline approaches, achieving 98.91% accuracy and 0.9894% score of ROC AUC using an 80:20 data split for training and testing. Additionally, performance metrics such as precision, recall, F1-score, and the ROC curve further validated the model's robustness in identifying botnet attacks.

The successful implementation of our model through the Streamlit web framework provides a user-friendly and accessible platform for real-time attack detection in IoT environments. Future work can focus on improving model generalization to unseen threats, incorporating federated learning for decentralized security, and optimizing computational efficiency for real-time deployment in resource-constrained IoT devices.

REFERENCES

[1] M. Ali, M. Shahroz, M. F. Mushtaq, S. Alfarhood, M. Safran, I. Ashraf, "Stacked Machine Learning Model for Efficient Botnet Attack Detection inIoT Environment," IEEE Access, vol. 12, pp. 40682-40699, 2024.

[2] M. Al-Fawa'reh, J. Abu-Khalaf, P. Szewczyk and J. J. Kang, "MalBoT-DRL: Malware Botnet Detection Using Deep Reinforcement Learning in IoT Networks,"IEEE Internet of Things Journal, vol. 11, no. 6, pp. 9610-9629, 15 March15, 2024.

[3] F. Taher, M. Abdel-Salam, M. Elhoseny and I. M. El-Hasnony, "Reliable Machine Learning Model for IIoT Botnet Detection,"IEEE Access, vol. 11, pp. 49319-49336, 2023.

[4] M. S. Alshehri, J. Ahmad, S. Almakdi, M. A. Qathrady, Y. Y. Ghadi, W. J. Buchanan, "SkipGateNet: A Lightweight CNN-LSTM Stacked Model with Learnable Skip Connections for Efficient Botnet Attack Detection in IoT," IEEE Access, vol. 12, pp. 35521-35538, 2024.

[5] R. Kalakoti, S. Nõmm and H. Bahsi, "In-Depth Feature Selection for the Statistical Machine Learning-Based Botnet Detection in IoT Networks," IEEE Access, vol. 10, pp. 94518-94535, 2022.

[6] D. Arnold, M. Gromov and J. Saniie, "Network Traffic Visualization Coupled with Convolutional Neural Networks for Enhanced IoT Botnet Detection," IEEE Access, vol. 12, pp. 73547-73560, 2024.

[7] M. Lefoane, I. Ghafir, S. Kabir, I. -U. Awan, K. El Hindi, A. Mahendran, "Latent Semantic Analysis and Graph Theory for Alert Correlation: A Proposed Approach for IoT Botnet Detection," in IEEE Open Journal of the Communications Society, vol. 5, pp. 3904-3919, 2024.

[8] S. A. Abdulkareem, C. Heng Foh, M. Shojafar, F. Carrez, K. Moessner, "Network Intrusion Detection: An IoT and Non IoT-Related Survey, "IEEE Access, vol. 12, pp. 147167-147191, 2024.

[9] E. Gelenbe, M. Nakıp, "Traffic Based Sequential Learning During Botnet Attacks to Identify Compromised IoT Devices, "IEEE Access, vol. 10, pp. 126536-126549, 2022.

[10] F. Sattari, A. H. Farooqi, Z. Qadir, B. Raza, H. Nazari, M. Almutiry, "A Stacked Deep Learning Approach for Bottleneck Detection in IoT, "IEEE Access, vol. 10, pp. 77039-77053, 2022.

[11] E. Gelenbe, M. Nakip, "IoT Network Cybersecurity Assessment with the Associated Random Neural Network, "IEEE Access, vol. 11, pp. 85501-85512, 2023.

[12] R. Dhakal, W. Raza, V. Tummala, L. Niure Kandel, "Enhancing Intrusion Detection in IoT Networks Through Federated Learning, "IEEE Access, vol. 12, pp. 167168-167182, 2024.

[13] O. M. Almorabea, T. J. S. Khanzada, M. A. Aslam, F. A. Hendi, A. M. Almorabea, "IoT Network-Based Intrusion Detection Framework: A Solution to Process Ping Floods Originating from Embedded Devices, "IEEE Access, vol. 11, pp. 119118-119145, 2023.

APPENDIX A1

SOURCECODE

```
import numpy as np
import pandas as pd
import joblib
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import seaborn as sns
# Load dataset
df = pd.read_csv('data_1.csv')
print(df)
# Preprocessing
dataframes = []
samples_per_file = 1999
normal_samples = df[df['category'] == 'Normal']
non_normal_samples = df[df['category'] != 'Normal']
random_samples = non_normal_samples.sample(n=samples_per_file,
random_state=42)
merged_df = pd.concat([normal_samples, random_samples])
dataframes.append(merged_df)
botiot = pd.concat(dataframes, ignore_index=True)
# Adjust category values
botiot.loc[botiot['category'] != 'Normal', 'category'] = 'Attack'
botiot.loc[botiot['category'] == 'Normal', 'category'] = '0_Normal'
# Drop irrelevant columns
botiot = botiot.drop(columns=['pkSeqID', 'stime', 'ltime', 'seq', 'smac', 'dmac',
'soui', 'doui', 'sco', 'dco', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'srate', 'drate', 'attack'])
botiot = botiot.dropna(subset=['sport'])
```

```
botiot.drop_duplicates(inplace=True)
if 'state' in botiot.columns:
    botiot = botiot.drop(columns=['saddr', 'daddr', 'state', 'sport', 'dport'])
else:
    botiot = botiot.drop(columns=['saddr', 'daddr', 'sport', 'dport'])
# Data Transformation using Label Encoding
label_encoder = LabelEncoder()
label_encoder_proto = LabelEncoder()
label_encoder_flgs = LabelEncoder()
botiot['proto'] = label_encoder_proto.fit_transform(botiot['proto'])
botiot['flgs'] = label_encoder_flgs.fit_transform(botiot['flgs'])
# Save encoders separately
joblib.dump(label_encoder_proto, 'label_encoder_proto.pkl')
joblib.dump(label_encoder_flgs, 'label_encoder_flgs.pkl')
# Check the mapping
proto_mapping = dict(zip(label_encoder_proto.classes_,
label_encoder_proto.transform(label_encoder_proto.classes_)))
flgs_mapping = dict(zip(label_encoder_flgs.classes_,
label_encoder_flgs.transform(label_encoder_flgs.classes_)))
print("Proto Encoding:", proto_mapping)
print("Flgs Encoding:", flgs_mapping)
# Separate features and target variable
botiot = botiot.drop(columns=['subcategory '])
X = botiot.drop(columns=['category'])
y = botiot['category']
# Encode target variable
y = y.map({'0_Normal': 0, 'Attack': 1}).astype(int)
# Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Save scaler
joblib.dump(scaler, 'scaler.pkl')
# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
```

```
random_state=42)
# Reshape for BiLSTM
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
#stacked DNN-BiLSTM Model
dnn_input = layers.Input(shape=(X_train.shape[1], X_train.shape[2]))
dnn = layers.Dense(512, activation='relu')(dnn_input)
dnn = layers.Dropout(0.3)(dnn)
dnn = layers.Dense(256, activation='relu')(dnn)
dnn = layers.Dropout(0.3)(dnn)
dnn = layers.Dense(128, activation='relu')(dnn)
dnn = layers.Dropout(0.3)(dnn)
dnn = layers.Dense(64, activation='relu')(dnn)
dnn = layers.Dropout(0.3)(dnn)
dnn = layers.Dense(32, activation='relu')(dnn)
bilstm_input = layers.Input(shape=(X_train.shape[1], X_train.shape[2]))
bilstm = layers.Bidirectional(layers.LSTM(50,
return_sequences=False))(bilstm_input)
bilstm = layers.Dense(32, activation='relu')(bilstm)
bilstm = layers.Dense(32, activation='relu')(bilstm)
dnn = layers.Reshape((32,))(dnn)
combined = layers.concatenate([dnn, bilstm])
output = layers.Dense(1, activation='sigmoid')(combined)
model = models.Model(inputs=[dnn_input, bilstm_input], outputs=output)
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Train model
history = model.fit([X_train, X_train], y_train, epochs=25, batch_size=256)
# Save model
model.save('botnet_model.h5')
# Model Evaluation
y_pred = model.predict([X_test, X_test])
y_pred_binary = (y_pred > 0.5).astype(int)
# Confusion Matrix
```

```
cm = confusion_matrix(y_test, y_pred_binary)
plt.figure(figsize=(7, 4))
lang = ['Normal', 'Attack']
cm = pd.DataFrame(cm, columns=lang, index=lang)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.title('Confusion Matrix - Test Set')
plt.xlabel('Predicted')
plt.ylabel('Actual')
# Accuracy and Classification Report
print(f"Accuracy: {accuracy_score(y_test, y_pred_binary) * 100:.2f}%")
print(classification_report(y_test, y_pred_binary, digits=4))
plt.show()
```

Botnet Attack Detection Web Interface

```
import base64
import streamlit as st
import pandas as pd
import numpy as np
import joblib
from tensorflow.keras.models import load_model
# Background Image Function
def get_base64(bin_file):
    try:
        with open(bin_file, "rb") as f:
            data = f.read()
        return base64.b64encode(data).decode()
    except FileNotFoundError:
        st.warning("ı Background image not found. Using default background.")
        return None
def set_background(png_file):
    bin_str = get_base64(png_file)
    if bin_str:
        page_bg_img = f"""
        <style>
```

```
        .stApp {{
            background-image: url("{bin_str}");
            background-size: cover;
        }}
        </style>
 st.markdown(page_bg_img, unsafe_allow_html=True)
# Set background image (optional)
set_background("1.jpg")
# Load saved model and preprocessing tools
model = load_model('botnet_model.h5')
label_encoder_proto = joblib.load('label_encoder_proto.pkl')
label_encoder_flgs = joblib.load('label_encoder_flgs.pkl')
scaler = joblib.load('scaler.pkl')
# Get the mapping of original labels to encoded values
proto_mapping = dict(zip(label_encoder_proto.classes_,
label_encoder_proto.transform(label_encoder_proto.classes_)))
flgs_mapping = dict(zip(label_encoder_flgs.classes_,
label_encoder_flgs.transform(label_encoder_flgs.classes_)))
# Title
title = '<h3 style="font-family:Segoe UI Black; color:blue; font-size: 32px;"> IoT
Botnet Attack Detection using Stacked Deep Learning Model (Customized DNN-
BiLSTM) </h3>'
st.markdown(title, unsafe_allow_html=True)
# File uploader
uploaded_file = st.file_uploader("Upload CSV File", type=["csv"])
if uploaded_file is not None:
    try:
        # Read the uploaded CSV
        df = pd.read_csv(uploaded_file)
        st.write("📊 Uploaded Data Preview:")
        st.dataframe(df.head())
        # Drop irrelevant or unnecessary columns (must match training
preprocessing)
```

```
        drop_cols = ['pkSeqID', 'stime', 'ltime', 'seq', 'smac', 'dmac', 'soui', 'doui',
'sco', 'dco',
                     'spkts', 'dpkts', 'sbytes', 'dbytes', 'srate', 'drate', 'attack',
                     'saddr', 'daddr', 'sport', 'dport']
        df = df.drop(columns=[col for col in drop_cols if col in df.columns],
errors='ignore')
        # Handle missing values
        if 'sport' in df.columns:
            df = df.dropna(subset=['sport'])
        if 'state' in df.columns:
            df = df.drop(columns=['state']
        # Label encode 'proto' and 'flgs'
        df['proto'] = label_encoder_proto.transform(df['proto'])
        df['flgs'] = label_encoder_flgs.transform(df['flgs'])
        # Normalize features
        print(df)
        X_scaled = scaler.transform(df)
        # Reshape for BiLSTM
        X_input = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))
        # Prediction button
        if  st.button('Q  Predict'):
            predictions = model.predict([X_input, X_input])
            print(predictions)
            binary_preds = (predictions > 0.5).astype(int).flatten()
            label_map = {0: 'Normal Traffic', 1: 'Botnet Attack'}
            results = [label_map[pred] for pred in binary_preds]  # Map predictions to
labels
            # Ensure results exist before accessing
            if len(results) > 0:
                st.markdown('<h3 style="color:white;">Q Predicted Result using
Customized DNN-BiLSTM:</h3>', unsafe_allow_html=True)
                st.markdown(f'<h3 style="color:darkblue;">{results[0]}</h3>',
unsafe_allow_html=True)
```

```
            else:
                st.error("ı No predictions were made. Check input data.")
            # Download option
            df['Prediction'] = results  # Store predictions in DataFrame
            csv_output = df.to_csv(index=False).encode('utf-8')
            st.download_button(label="📥 Download Results as CSV",
data=csv_output, file_name="botnet_predictions.csv", mime='text/csv')
    except Exception as e:
        st.error(f"ı An error occurred during processing: {e}")
```

APPENDIX A2

SCREENSHOTS

	pkSeqID	stime	flgs	proto	saddr	sport	daddr	dport	pkts	bytes	...	spkts	dpkts	sbytes	dbytes	rate	srate	drate	attack	category	subcategory
0	1	1.526344e+09	e	arp	192.168.100.1	NaN	192.168.100.3	NaN	4	240	...	2	2	120	120	0.002508	0.000836	0.000836	0	Normal	Normal
1	2	1.526344e+09	e	tcp	192.168.100.7	139	192.168.100.4	36390	10	680	...	5	5	350	330	0.006190	0.002751	0.002751	0	Normal	Normal
2	3	1.526344e+09	e	udp	192.168.100.149	51838	27.124.125.250	123	2	180	...	1	1	90	90	20.590960	0.000000	0.000000	0	Normal	Normal
3	4	1.526344e+09	e	arp	192.168.100.4	NaN	192.168.100.7	NaN	10	510	...	5	5	210	300	0.006189	0.002751	0.002751	0	Normal	Normal
4	5	1.526344e+09	e	udp	192.168.100.27	58999	192.168.100.1	53	4	630	...	2	2	174	456	0.005264	0.001755	0.001755	0	Normal	Normal
5	6	1.526344e+09	e	arp	192.168.100.1	NaN	192.168.100.27	NaN	2	120	...	1	1	60	60	2724.795654	0.000000	0.000000	0	Normal	Normal
6	7	1.526344e+09	e	arp	192.168.100.27	NaN	192.168.100.1	NaN	4	240	...	2	2	120	120	0.005264	0.001755	0.001755	0	Normal	Normal
7	8	1.526344e+09	e	udp	192.168.100.150	58360	192.168.217.2	53	2	172	...	2	0	172	0	0.399984	0.399984	0.000000	0	Normal	Normal
8	9	1.526344e+09	e	udp	192.168.100.149	37214	192.168.217.2	53	2	172	...	2	0	172	0	0.399824	0.399824	0.000000	0	Normal	Normal
9	10	1.526344e+09	e	arp	192.168.100.150	NaN	192.168.100.1	NaN	6	360	...	3	3	180	180	0.004339	0.001736	0.001736	0	Normal	Normal

Fig. A2.1 Dataset Loading

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 1, 11)	0	-
dense (Dense)	(None, 1, 512)	6,144	input_layer[0][0]
dropout (Dropout)	(None, 1, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 1, 256)	131,328	dropout[0][0]
dropout_1 (Dropout)	(None, 1, 256)	0	dense_1[0][0]
dense_2 (Dense)	(None, 1, 128)	32,896	dropout_1[0][0]
dropout_2 (Dropout)	(None, 1, 128)	0	dense_2[0][0]
dense_3 (Dense)	(None, 1, 64)	8,256	dropout_2[0][0]
input_layer_1 (InputLayer)	(None, 1, 11)	0	-
dropout_3 (Dropout)	(None, 1, 64)	0	dense_3[0][0]
bidirectional (Bidirectional)	(None, 100)	24,800	input_layer_1[0][0]
dense_4 (Dense)	(None, 1, 32)	2,080	dropout_3[0][0]
dense_5 (Dense)	(None, 32)	3,232	bidirectional[0][0]
reshape (Reshape)	(None, 32)	0	dense_4[0][0]
dense_6 (Dense)	(None, 32)	1,056	dense_5[0][0]
concatenate (Concatenate)	(None, 64)	0	reshape[0][0], dense_6[0][0]
dense_7 (Dense)	(None, 1)	65	concatenate[0][0]

Total params: 209,857 (819.75 KB)
Trainable params: 209,857 (819.75 KB)
Non-trainable params: 0 (0.00 B)

Fig. A2.2 Stacked Model Building and Training

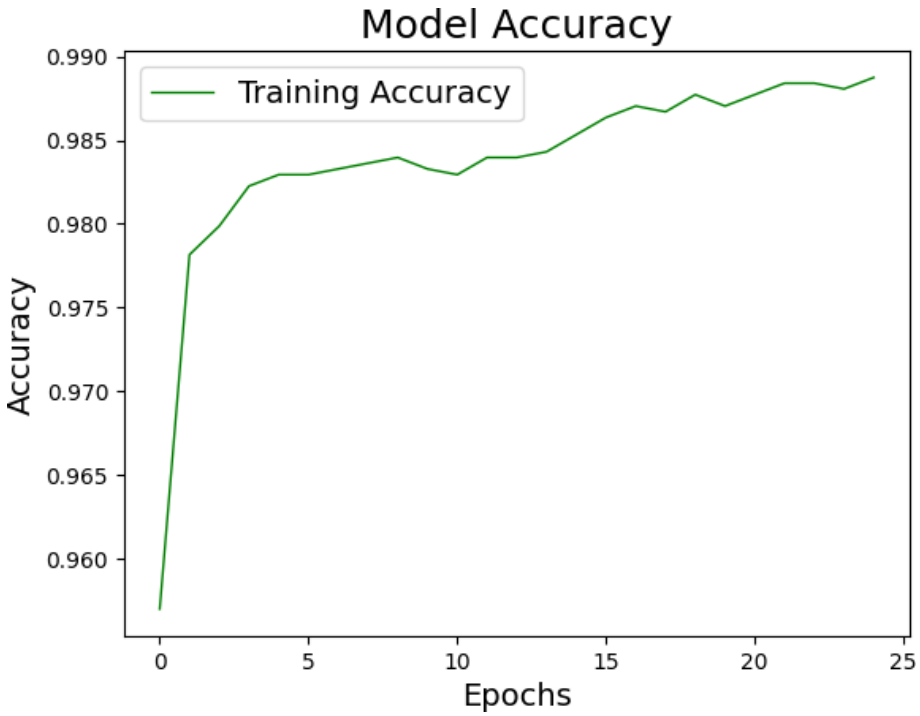

Fig. A2.3 Training Performance

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step
Predicted Results:
array([['Attack'],
       ['Attack'],
       ['Attack'],
       ['Attack'],
       ['Attack'],
       ['Normal'],
       ['Normal'],
       ['Attack'],
       ['Attack'],
       ['Attack'],
       ['Normal'],
       ['Normal'],
       ['Normal'],
       ['Attack'],
       ['Attack'],
       ['Attack'],
       ['Normal'],
       ['Normal'],
       ['Attack'],
       ['Normal'],
       ['Normal'],
       ['Normal'],
```

Fig. A2.4 Testing Set Result based on Trained DNN-BiLSTM Model

Accuracy on test set: 98.91%

	precision	recall	f1-score	support
0	0.9828	0.9942	0.9885	345
1	0.9948	0.9845	0.9896	388
accuracy			0.9891	733
macro avg	0.9888	0.9894	0.9891	733
weighted avg	0.9892	0.9891	0.9891	733

Fig. A2.5 Performance Metrics

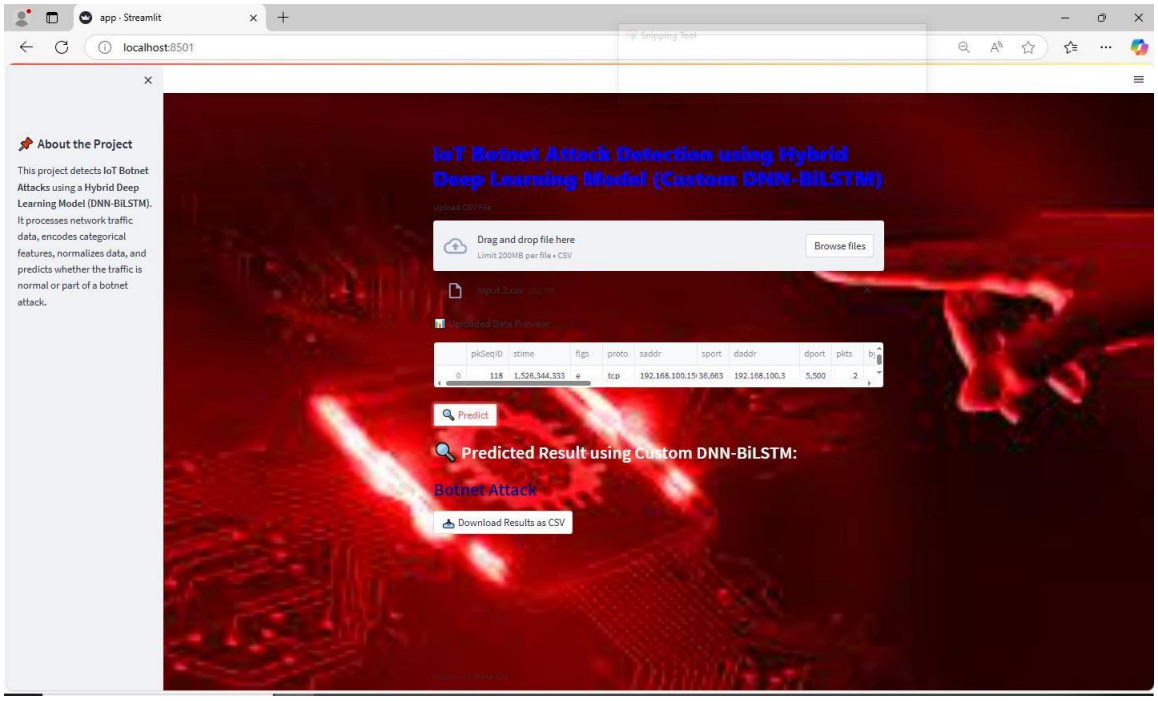

Fig. A2.6 Web Interface Attack Detection

TECHNICAL BIOGRAPHY

Mr. Aravind A (231272601001) is a Second Year Master of Technology (M. Tech) student specializing in Information Technology at B.S. Abdur Rahman Crescent Institute of Science and Technology, Tamil Nadu, India. He has completed his B. Tech Degree in Information Technology from Adhiparasakthi Engineering College, Tamil Nadu, India. His areas of interest include Machine learning and Data Science, with a strong aspiration to become a proficient Data Analyst and has good skills in Python, R Programming, Data Visualization, etc.

PLAGIARISM REPORT

Project Document (2)

ORIGINALITY REPORT

7%	3%	7%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	1%
2	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	1%
3	Mudasir Ali, Mobeen Shahroz, Muhammad Faheem Mushtaq, Sultan Alfarhood, Mejdl Safran, Imran Ashraf. "Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment", IEEE Access, 2024 Publication	1%
4	Singh, Gurvinder. "Real-Time Quantum Computing Anomaly Detection Model on Vulnerabilities in Government Systems.", The George Washington University Publication	<1%
5	doaj.org Internet Source	<1%
6	"Internet Computing and IoT and Embedded Systems, Cyber-physical Systems, and Applications", Springer Science and Business Media LLC, 2025 Publication	<1%
7	Archana Kalidindi, Mahesh Babu Arrama. "Feature selection and hybrid CNNF deep	<1%

	stacked autoencoder for botnet attack detection in IoT", Computers and Electrical Engineering, 2025 Publication	
8	ro.ecu.edu.au Internet Source	<1 %
9	Al-Sakib Khan Pathan. "The State of the Art in Intrusion Prevention and Detection", Auerbach Publications, 2019 Publication	<1 %
10	Submitted to Texas State University- San Marcos Student Paper	<1 %
11	discovery.researcher.life Internet Source	<1 %
12	ijaers.com Internet Source	<1 %
13	napier-repository.worktribe.com Internet Source	<1 %
14	tojqi.net Internet Source	<1 %
15	www.mdpi.com Internet Source	<1 %
16	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P. Prasad. "Algorithms in Advanced Artificial Intelligence - Proceedings of International Conference on Algorithms in Advanced Artificial Intelligence (ICAAAI-2024)", CRC Press, 2025 Publication	<1 %
17	eprints.intimal.edu.my Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		