A Comprehensive Study of Deep Learning Models for Intrusion Detection in IoT Devices

Enas F. Khairallah
Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia
ekhairallah@kau.edu.sa (corresponding author)

Nibras Alsenani
Department of Computer Science, Applied College, Northern Border University, Saudi Arabia | Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia
nibras.alsenani@nbu.edu.sa

Received: 3 November 2024 | Revised: 29 December 2024 | Accepted: 18 January 2025

 DOI: https://doi.org/10.48084/etasr.9490

ABSTRACT

The Internet of Things (IoT) has revolutionized how people interact with the world, but the increasing complexity of cyberattacks poses significant challenges in detecting intrusions. Failure to prevent intrusions can compromise IoT security services, including data confidentiality, integrity, and availability. For this reason, this study employs four deep learning models: A Deep Neural Networks (DNN), a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN), and a Long-Short-Term Memory (LSTM) network. The multiclassification performance of each model was evaluated using the Bot-IoT dataset. This study also addresses the bias towards the DDoS/DoS category in the Bot-IoT dataset, using the SMOTE technique to mitigate overfitting. The LSTM model achieved an excellent balance between performance and efficiency, outperforming state-of-the-art deep learning Intrusion Detection System (IDS) approaches on the same dataset, achieving a multiclass classification accuracy of 99.97%.

Keywords-deep neural network; intrusion detection system; deep learning; internet of things; cybersecurity

I. INTRODUCTION

The Internet of Things (IoT) is a rapidly expanding network of smart devices, such as home appliances, sensors, mobile phones, and computers, projected to reach 50 billion devices by 2020 [1]. This proliferation has made the IoT a cornerstone of modern technology, enabling smart cities, homes, and various application domains that enhance productivity and convenience [2]. However, the widespread adoption of IoT devices also exposes them to significant security threats, particularly in critical sectors such as healthcare and energy, where attacks such as Denial-of-Service (DoS) and Distributed DoS (DDoS) can have devastating consequences. The interconnected and resource-constrained nature of IoT devices poses unique challenges in implementing robust security measures. Although securing data transmission between IoT devices is essential, many devices lack the computational and energy capacity to support advanced security features [1, 3]. Furthermore, IoT networks often face novel and evolving attacks, exacerbating their vulnerability to cyber threats [4]. Addressing these challenges requires innovative approaches to intrusion detection that balance effectiveness with the resource limitations of IoT systems.

Recent advances in Artificial Intelligence (AI), particularly Deep Learning (DL), have shown promise in addressing these challenges. DL algorithms excel in processing heterogeneous data, detecting dependencies, and learning attack patterns to enhance Intrusion Detection Systems (IDSs) [5]. However, heavy computational tasks, such as model training and big data analysis, often need to be outsourced to fog or cloud servers due to the limited computational capabilities of IoT devices [6]. Although this offload reduces execution delays and power consumption, it also introduces new security concerns [2].

Despite progress in IDSs for IoT networks, several challenges persist. IoT intrusion datasets are often imbalanced, with benign activities dominating rare intrusion types, making it difficult to detect anomalies accurately. IoT devices face diverse intrusion patterns that are easier to classify with sophisticated techniques. To address these challenges, this study proposes a deep learning-based approach that leverages the Synthetic Minority Oversampling Technique (SMOTE) for dataset balancing and evaluates the performance of multiple DL models, including DNN, CNN, RNN, and LSTM, to classify IoT network activities as benign or malicious.

A. Motivation

IDSs are a critical line of defense for securing IoT networks against cyber threats. DL-based IDSs can effectively learn benign and anomalous behavior patterns in IoT networks and provide robust security protocols. This study aims to enhance the detection of rare intrusion types while maintaining efficiency and scalability for resource-constrained IoT systems.

B. Research Problem

Current IDSs face limitations due to imbalanced datasets, resource constraints, and the diverse nature of IoT network threats. This study seeks to overcome these challenges by developing DL algorithms tailored for IoT intrusion detection, utilizing SMOTE for dataset balancing, and testing the performance of various DL architectures for accuracy, efficiency, and scalability.

C. Literature Review

IoT networks allow large-scale data transfers between devices but are vulnerable to attacks that compromise confidentiality, integrity, and availability. The challenge of building an IDS for IoT lies in the large amounts of data that need real-time analysis. In [7], a deep ensemble-based IDS was introduced, employing the Lambda architecture with LSTM, CNN, and ANN classifiers. The batch layer trained the model, while the speed layer enhanced real-time decision-making. The hybrid ensemble achieved 99.93% accuracy for multiclass classification, outperforming individual classifiers in processing time and accuracy. In [8], a Fully Connected Feed Forward Neural Network (FCFFN) was proposed to detect malicious traffic. The four-layer deep neural network architecture achieved a 93.71% detection rate but was limited to detecting five types of intrusions from an experimental dataset. In [9], a hybrid IDS framework was proposed, using LSTM, CNN, and CNN-LSTM models with the IoTID20 dataset and Particle Swarm Optimization (PSO) for feature selection.

In [10], anomaly-based IDSs using DL approaches in IoT environments were reviewed. This study introduced methods leveraging ML techniques to detect threats but highlighted issues with false-positive rates and power consumption in IoT devices. This review provides a foundation for understanding DL approaches to IoT security. In [11], a comprehensive review of IoT systems was presented, including protocols, architecture, risks, and IDS strategies. This study highlighted the difficulties in designing an IDS model that balances accuracy, scalability, and comprehensive attack protection. In [2], recent IDSs for IoT were reviewed, focusing on methods, features, and procedures. However, this study lacked emphasis on the IoT architecture and DL-based IDS models. In [12], feature extraction and selection methods were proposed for IDS using Swarm Intelligence (SI) algorithms. This study introduced a CNN-based feature extraction system and an Aquila optimizer-based feature selection strategy but lacked detailed experimental results. In [13], a deep learning-based IDS was proposed for IoT networks, using a feed-forward neural model. Although this model achieved accurate results for binary classification, it was inadequate for multiclass classification.

II. METHODOLOGY

To achieve security and control in the IoT infrastructure, this study focuses on DL algorithms to detect and recognize attacks in real time. The Bot-IoT dataset was created to simulate a realistic network environment at the Canberra Cyber Range Lab of the University of New South Wales (UNSW) [14, 15]. It is a comprehensive and labeled dataset tailored for cybersecurity research, focusing on IoT network traffic patterns. The dataset includes normal IoT activities and botnet intrusion attacks, making it valuable for developing and evaluating IDSs and network forensic analysis techniques. The Bot-IoT dataset consists of more than 72 million records. It features a wide range of IoT network traffic flows, with simulated botnet attacks of different intrusion types, including DoS, DDoS, OS, keylogging, data exfiltration attacks, service scans, and regular communication patterns. This diversity makes it especially useful for training machine learning models, detecting network anomalies, and enhancing security measures for IoT networks. This dataset has become a key resource in IoT cybersecurity research. Each row has 46 features, and each record is assigned benign or malicious activity. As shown in Table I, there are 10 subcategories of attacks.

TABLE I. DATASET STATISTICS

Category	Subcategory	Number of records
DDoS	UDP	576884
	TCP	46038
	HTTP	989
DoS	TCP	212513
	UDP	37344
	HTTP	1485
Normal	Normal	477
Reconnaissance	Service_Scan	73168
	OS_Fingerprint	17914
Theft	Keylogging	73
	Data_Exfiltration	6

As the full dataset is extremely large (~16.7 GB), this study utilized a subset containing 5% of the total dataset. Although 5% of 73 million records should be approximately 3.65 million, the selected dataset consists of 966,829 records. This discrepancy is due to data preprocessing and filtering techniques, which ensured that only clean and representative records were retained for model training and evaluation. To construct the subset, records were extracted from different Bot-IoT dataset files to form a comprehensive CSV file containing both benign and attack traffic. During filtering and preprocessing, redundant and corrupted records were removed to enhance data quality, and a balanced proportion of various attack categories and normal activity was retained to improve model performance. The final dataset composition included 477 benign records and 966,414 attack records, distributed across DDoS, DoS, reconnaissance, and theft categories. Table I presents the statistics of the dataset used in this study.

This study applied four DL models, namely, a Deep Neural Network (DNN), a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN), and a Long Short-Term Memory (LSTM) network.

A. Deep Neural Network (DNN)

DNNs are commonly employed to handle complicated problems in machine and deep learning. An artificial neuron is the most fundamental component of a DNN, inspired by biological neurons in the human brain [16]. This study used Multilayer Perceptrons (MLPs), a form of Feed-Forward Deep Neural Network (FFDNN) with more than three layers. The information flows in FFDNN in just one way: from the input layers to the output layers via the hidden layers. A hyperparameter selection defines the number of hidden layers [17]. The Deep Neural Network architecture consists of an input layer with 35 features, followed by two hidden layers with 64 neurons each and ReLU activation, and an additional hidden layer with 32 neurons using ReLU. The output layer contains 5 neurons with a Softmax activation function for multiclass classification. Fully connected layers ensure that each neuron connects to every neuron in the next layer, allowing hierarchical feature extraction. The ReLU activation function enhances learning in hidden layers, while Softmax converts the final output into class probabilities [17].

B. Convolutional Neural Network (CNN)

A CNN captures high-resolution data before converting it into complex features at a lower resolution. CNNs consist of three main types of layers: convolutional, pooling, and fully connected layers. Convolution and pooling layers extract features, while fully connected layers process them for classification [18]. The CNN architecture begins with an input layer that processes 35 features reshaped as a 1D input. A convolutional layer with 32 filters, a kernel size of 3, and ReLU activation extracts spatial patterns. A flatten layer converts the 2D output into a 1D format, followed by two fully connected hidden layers with 64 neurons each and an additional hidden layer with 32 neurons, all using ReLU activation. The output layer consists of 5 neurons with a Softmax activation function for multiclass classification. CNN leverages convolutional filters for hierarchical feature extraction, while fully connected layers refine the classification process.

C. Recurrent Neural Network (RNN)

An RNN is a type of neural network in which connections form cycles, allowing information to persist over time steps. RNNs process sequential data by incorporating an interconnection matrix to capture temporal dependencies [19]. The architecture includes an input layer with 35 neurons, followed by a SimpleRNN layer with 64 neurons for sequence processing. Two fully connected hidden layers with 64 and 32 neurons, both using ReLU activation, refine the extracted features. The output layer consists of 5 neurons with a Softmax activation function for classification. Recurrent connections enable learning from sequential patterns in time-series data.

D. Long Short-Term Memory (LSTM)

LSTM networks are an advanced form of RNN designed to address the vanishing gradient problem, allowing data to persist over long sequences. LSTMs consist of three functional gates: the forget gate determines whether past information should be retained or discarded, the input gate processes new information, and the output gate updates the current state [20]. LSTM begins with an input layer processing 35 features reshaped as a 1D input, followed by a Conv1D layer with 32 filters and ReLU activation to extract features. An LSTM layer with 32 units maintains temporal dependencies. A flatten layer converts the multidimensional output into a 1D format, followed by two fully connected hidden layers with 64 neurons each and an additional hidden with 32 neurons, all using ReLU activation. The final output layer consists of 5 neurons with Softmax activation for classification. This structure integrates spatial and temporal feature extraction for improved efficiency.

E. Proposed Model of IDS-based DL Structure

The proposed model for IoT IDS is based on deep learning, building on the architectures of DNN, CNN, RNN, and LSTM to classify network traffic into benign or malicious. The model consists of three main stages: (i) preprocessing, (ii) training, and (iii) testing. In the preprocessing stage, the dataset is cleaned, normalized, and balanced using techniques such as SMOTE to address class imbalance. The model is then trained on 60% of the dataset, with 20% allocated for validation, to ensure that the model generalizes well to unseen data. In the training stage, the network is trained using the training data, and optimization techniques such as the Adam optimizer and cross-entropy loss are employed for efficient learning and error minimization. Finally, in the testing stage, the trained model is evaluated on the remaining 20% of the dataset to assess its performance in detecting malicious and benign traffic.

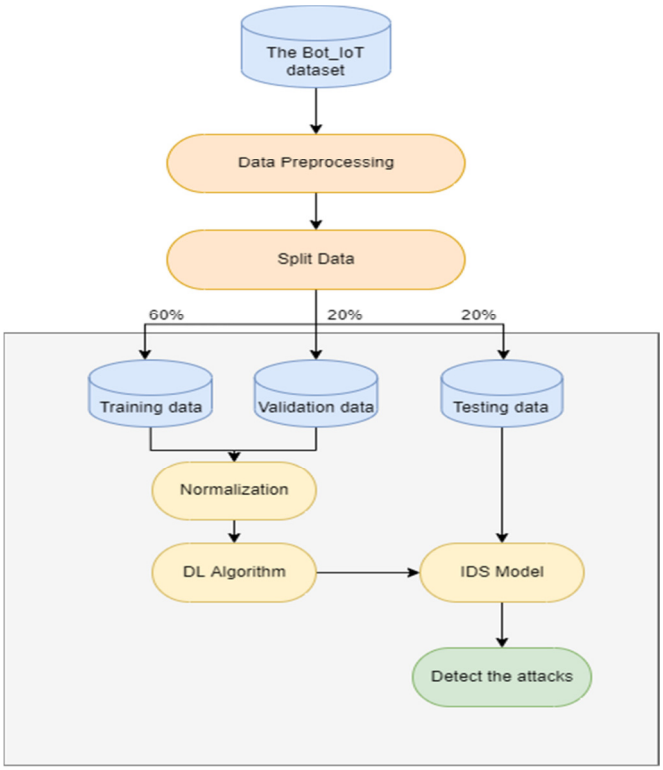

Fig. 1. Proposed IDS DL structure.

The model architecture consists of an input layer, several fully connected hidden layers, and an output layer. The input

layer processes 46 features extracted from the Bot-IoT dataset, representing various network traffic characteristics. The hidden layers include two dense layers: the first with 64 neurons and the second with 32 neurons, designed to capture complex patterns in the network traffic data. A dropout layer with a rate of 0.2 is applied after each dense layer to prevent overfitting and ensure that the model generalizes well. The output layer consists of five neurons with a softmax activation function, producing multiclass classification results for the five types of network attacks in the dataset.

The model utilizes fully connected dense layers, where each neuron in one layer is connected to all neurons in the next layer. For the RNN and LSTM components, temporal connections are employed to capture the sequential nature of network traffic patterns over time, which is essential for detecting time-dependent anomalies in IoT traffic. Feature extraction is accomplished through dense layers for hierarchical feature representation, and if CNN layers are used, they assist in spatial feature detection. LSTM layers specifically address long-term dependencies in IoT network traffic and mitigate issues such as vanishing gradients that are common in standard RNNs. The hidden layers use ReLU activation functions to introduce nonlinearity, and the output layer applies Softmax for multiclass classification.

For training, the Adam optimizer was used, and cross-entropy loss was used to compute the error between the predicted and actual attack categories. The model was trained over 50 epochs with a batch size of 1024, and GPU acceleration was used to speed up the computation process. TensorFlow was employed to build and test the IDS model.

III. IMPLEMENTATION

A 2.3 GHz dual-core Intel Core i5 processor with 8 GB RAM was used to implement the proposed model. Regarding software, the model was run on Google Collab, utilizing TensorFlow and Graphics Processing Unit under Python 3. Furthermore, the Keras API, Scikit-learn, Panda, and Seaborn libraries were used.

A. Dataset Balancing

SMOTE was applied to address the class imbalance in the Bot-IoT dataset. SMOTE is an oversampling technique that generates synthetic samples for the minority classes by interpolating between existing data points, rather than simply duplicating records as in random oversampling. By focusing on the feature space and creating new samples through interpolation between close positive examples, SMOTE helps mitigate overfitting and ensures a more balanced dataset [21]. Before oversampling, the dataset contained 966,414 attack records and only 477 normal records, with more than 96% of the attack records belonging to the DoS and DDoS classes. Consequently, the model would accurately predict the majority classes while failing to correctly predict the minority classes, indicating model bias [22, 23]. To address this imbalance, SMOTE was applied to the normal, reconnaissance, and theft classes, increasing the sample size of each to 623,911 records, as shown in Table II. After this oversampling, the dataset became more balanced, allowing the model to predict both majority and minority classes more effectively.

TABLE II. DATASET RECORDS FOR ORIGINAL AND OVERSAMPLED DATA

Normal/Attack	Original dataset	Oversampled dataset
DDoS	623,911	623,911
DoS	251,342	623,911
Theft	79	623,911
Reconnaissance	91,082	623,911
Normal	477	623,911

B. Dataset Preprocessing

Preprocessing transforms raw data into a clean and structured format, improving model accuracy. This includes data cleansing, normalization, and feature transformation. Missing or incomplete values in the Bot-IoT dataset were removed using the pandas dropna() method. Superfluous features, such as *pkSeqID* (row identifier), *stime* and *ltime* (captured in dur), and network flow identifiers (e.g., source and destination IPs), were eliminated. Relevant features, such as *packet_count*, *byte_count*, *packet_rate*, and *attack_labels*, were retained. Additionally, categorical features were transformed for computational efficiency, ensuring that the dataset was clean and ready for model training.

C. Feature Transformation

Feature transformation can boost model performance by applying a mathematical formula to a feature to convert values into a usable format. Categorical features, including *flgs_number*, *proto_number*, *sport*, *dport*, *state_number*, *attack*, *category*, and *subcategory*, were transformed into binary indicator features (0s and 1s) using the get_dummies() function in Python. Data standardization rescales attributes to a mean of 0 and a variance of 1, ensuring consistency without distorting value differences. As some features in the Bot-IoT dataset have varying ranges, making learning complex for models, this step helps achieve optimal performance.

D. Dataset Splitting

Cross-validation techniques are common methods for ensuring excellent generalization and avoiding overtraining. The goal is to divide the dataset into three sections. Two subsets are employed for training and validation, while the final model's performance is tested on the remaining subset. Cross-validation aims to arrive at a stable and reliable model performance estimate. Therefore, 42 is applied for the random state to seed the random generator so that it is always deterministic in the train and test splits. Using the traditional three-way split, the dataset was divided into 60%, 20%, and 20% subsets for training, validation, and testing, respectively.

E. Deep Learning Model

The model was built in Google Collab by applying the Keras API of TensorFlow. Preprocessing, model, layers, and optimizer were based on Keras packages. The ReLU activation function was used to model the hidden layers' nonlinear relationships. Softmax is an activation function, also called generalized logistic regression, employed at the output layer. In addition to the normal class, the output unit numbers employed in softmax are equal to the attack category numbers. Table III shows the hyperparameters used.

TABLE III. HYPERPARAMETERS USED

Hyperparameter	Value
Hidden Nodes (HN)	64
Optimizer	Adam
Activation function	ReLu
Classification function	Softmax
Number of epochs	50
Number of batches	1024
Dropout	.5

IV. RESULTS

A. Performance Metrics

Performance metrics indicate how well the detection model can distinguish different kinds of network traffic. The most essential performance indicators are accuracy, precision, recall, and F1-score.

- Accuracy represents the percentage of the correct predictions that were successfully identified. It is also the proportion of correct detections to total records in the dataset. Accuracy is calculated as follows:

$$\text{Acuracy} = \frac{\text{Tp+TN}}{\text{TP+TN+FP+FN}} \quad (1)$$

where FP denotes false positives, TP denotes true positives, FN represents false negatives, and FP represents false positives [24].

- Precision: This score describes the classifier's ability to predict normal data without any conditions [19].

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (2)$$

- Recall is the proportion of correctly classified records to the number of occurrences, calculated as [24]:

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

- F1-score is the harmonic mean of recall and precision, calculated as [24]:

$$\text{F1} - \text{score} = \frac{\text{2TP}}{\text{2TP+FP+FN}} \quad (4)$$

- Cross-entropy loss (log loss) evaluates the classification model's performance, which is expressed as a probability value. A good model would have a log loss of 0, which increases when the anticipated probability differs from the actual label.
- Training time denotes the time required to construct the categorization model.

B. Results

To achieve the best performance, a series of experiments was executed with various hyperparameter values (batch size, epochs, and number of layers). The best results were obtained using a large number of epochs (10, 20, 40, and 120) and several batch sizes (100, 512, and 1024). Increasing the number of epochs causes the model to run slower. Similarly, reducing the batch size did not enhance performance. The batch size and the number of epochs were both specified as 1024 and 50, respectively.

Figure 2 illustrates the accuracy of the oversampled dataset for multiclass classification in DNN, CNN, RNN, and LSTM during the training and validation stages. It can be noticed that DNN's accuracy was increased in the last 20 epochs, while the other three models achieved a higher performance in the first 10 epochs, indicating that 50 epochs would be enough. Furthermore, these results indicate no overfitting in the training and validation processes.

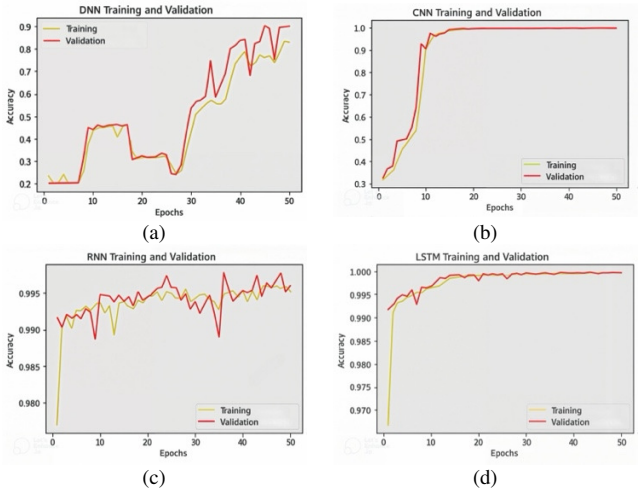

Fig. 2. Accuracy vs epochs for the four models: (a) DNN, (b) CNN, (c) RNN, (d) LSTM.

Figure 3 shows the log loss results of DNN, CNN, RNN, and LSTM, where it can be noticed that the LSTM model excels over the other three models.

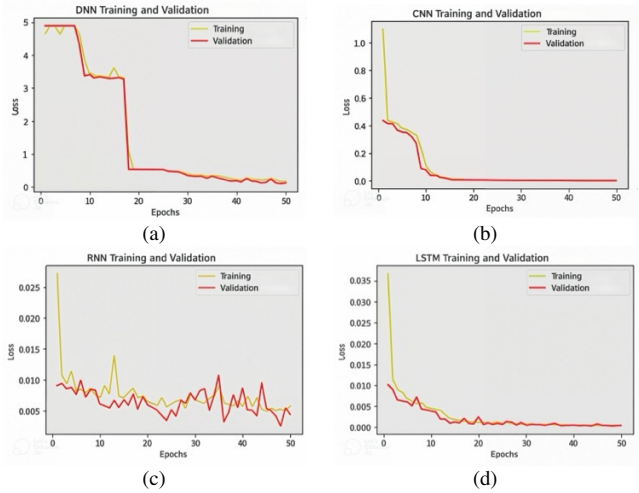

Fig. 3. Log loss vs epochs for the four models: (a) DNN, (b) CNN, (c) RNN, (d) LSTM.

Table IV and Figures 4 and 5 show the performance of the four models on the original dataset. RNN and LSTM achieved the highest accuracy, reaching 98.65% and 99.96%, respectively, while DNN and CNN showed lower performance. Even without dataset balancing, deep learning models

produced strong results. After applying SMOTE oversampling, CNN, RNN, and LSTM improved their accuracy to 99.81%, 99.60%, and 99.97%, respectively, while DNN experienced a slight reduction to 90.15%.

Oversampling significantly improved precision, recall, and F1-score, particularly in detecting the minority classes. In particular, CNN, RNN, and LSTM achieved 100% precision and recall after oversampling due to the balanced class distribution created by SMOTE. However, the slight reduction in overall accuracy compared to the original dataset may be attributed to overfitting the minority classes or the noise introduced during oversampling.

TABLE IV. PERFORMANCE OF THE MODELS

	Dataset type	DNN	CNN	RNN	LSTM
Accuracy	Original	91.32%	94.89%	98.65%	99.96%
	Oversampling	90.15%	99.81%	99.60%	99.97%
Precision	Original	75%	77%	79%	100%
	Oversampling	91%	100%	100%	100%
Recall	Original	54%	56%	75%	86%
	Oversampling	90%	100%	100%	100%
F1-score	Original	56%	57%	77%	89%
	Oversampling	90%	100%	100%	100%
Log loss	Original	12.13	6.82	1.60	0.06
	Oversampling	12.63	0.24	0.45	0.04
Training time	Original	20s 3ms	14s 2ms	0.06	20s 3ms
	Oversampling	29s 2ms	31s 2ms	0.04	126s 6ms

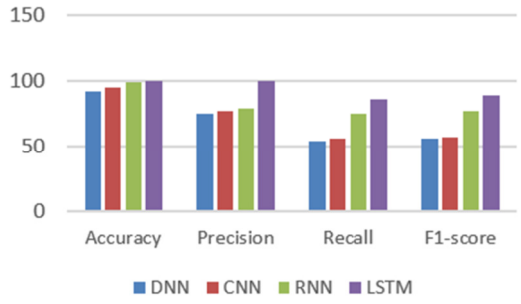

Fig. 4. Model performance chart (original dataset).

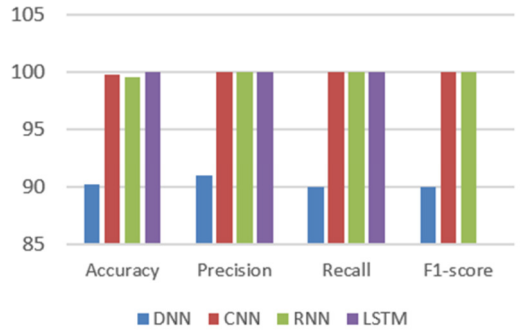

Fig. 5. Model performance chart (oversampled dataset).

Table V illustrates more details about the performance of each attack type in the Bot-IoT dataset, allowing comparisons between original versus oversampled datasets. The prominence of theft and normal category in the DNN and CNN models is striking in the oversampled dataset, achieving 89.75%, 99.98%, 94.59%, and 99.75%, 99.99%, and 99.87%, precision, recall, and F1-score for theft records, and 93.13%, 96.32%, 94.71%, and 99.97%, 99.79%, 99.88% for normal records, respectively. Unlike the original dataset, theft and normal categories record small values in DNN and CNN models. Moreover, the accuracy is high (91-99%).

For RNN and LSTM, the theft category has small values in the original dataset. After balancing the dataset, these two models hit all performance metrics by recording higher scores in all attack types (>98%). Significantly, the accuracy never decreased below 98% for RNN and LSTM after balancing the dataset.

TABLE V. PERFORMANCE ON ATTACK TYPES

Model	Dataset	Category	Accuracy	Precision	Recall	F1-score
DNN	Original	DDoS	92.38%	94.92%	93.17%	94.04%
		DoS	91.46%	81.71%	86.47%	84.02%
		Theft	99.99%	0	0	0
		Reconnaissance	98.74%	99.71%	86.89%	92.86%
		Normal	99.95%	66.66%	2.43%	4.7%
	Over-sampled	DDoS	95.30%	92.48%	83.29%	87.64%
		DoS	94.60%	82.40%	92.87%	87.32%
		Theft	97.71%	89.75%	99.98%	94.59%
		Reconnaissance	95.28%	99.23%	77.01%	86.72%
		Normal	97.85%	93.13%	96.32%	94.71%
CNN	Original	DDoS	95.46%	96.25%	96.74%	96.49%
		DoS	95.56%	90.26%	92.95%	91.58%
		Theft	99.99%	0	0	0
		Reconnaissance	98.77%	99.75%	87.26%	93.09%
		Normal	99.95%	1.0%	2.43%	4.76%
	Over-sampled	DDoS	99.72%	99.92%	98.69%	99.30%
		DoS	99.74%	98.99%	99.72%	99.63%
		Theft	99.95%	99.75%	99.99%	99.87%
		Reconnaissance	99.86%	99.49%	99.84%	99.66%
		Normal	99.95%	99.97%	99.79%	99.88%
RNN	Original	DDoS	98.68%	99.26%	98.68%	98.97%
		DoS	98.74%	96.87%	98.45%	97.61%
		Theft	99.99%	0	0	0
		Reconnaissance	99.98%	99.81%	99.12%	99.46%
		Normal	99.99%	95.71%	81.70%	88.15%
	Over-sampled	DDoS	99.72%	99.65%	98.95%	99.30%
		DoS	99.75%	98.94%	99.86%	99.40%
		Theft	99.95%	99.78%	99.97%	99.88%
		Reconnaissance	99.91%	99.98%	99.61%	99.79%
		Normal	99.95%	99.93%	99.86%	99.89%
LSTM	Original	DDoS	99.96%	99.98%	99.95%	99.97%
		DoS	99.96%	99.90%	99.95%	99.93%
		Theft	99.99%	1.0%	18.75%	31.75%
		Reconnaissance	99.98%	99.90%	99.98%	99.94%
		Normal	99.99%	98.76%	97.56%	98.15%
	Over-sampled	DDoS	99.98%	99.96%	99.97%	99.96%
		DoS	99.98%	99.97%	99.95%	99.96%
		Theft	99.98%	99.92%	100%	99.96%
		Reconnaissance	99.98%	99.99%	99.99%	99.99%
		Normal	99.98%	99.99%	99.93%	99.96%

C. Findings

Among the tested models, LSTM achieved the highest accuracy of 99.97%. The other models (CNN: 99.81%, RNN: 99.60%, DNN: 99.15%) performed well but were outperformed by LSTM due to its ability to capture temporal dependencies effectively. Balancing the dataset with SMOTE improved the detection of minority classes, significantly increasing precision, recall, and F1 scores. The effectiveness of SMOTE was evident

in detecting rare intrusion types, such as reconnaissance and theft, which had low detection rates in unbalanced datasets. The LSTM model demonstrated high performance with manageable computational costs, making it suitable for real-time IoT intrusion detection. LSTM outperformed DNN, CNN, and RNN in overall performance metrics while maintaining scalability and low latency for resource-constrained environments.

D. Strengths and Weaknesses

This study exhibits several strengths and some limitations that are important to highlight. Strengths include a comprehensive comparison of four deep learning models, providing valuable insights into their performance for intrusion detection in IoT networks. Implementing SMOTE to address dataset imbalance improved the detection of minority classes and addressed biases in the Bot-IoT dataset. Furthermore, using a real-world dataset such as Bot-IoT enhances the practical applicability of the findings. In particular, the LSTM model demonstrated superior performance, achieving 99.97% multiclass classification accuracy, highlighting its efficiency and robustness for intrusion detection tasks. However, this study has certain weaknesses. The analysis is limited in scope, primarily focusing on DDoS/DoS, theft, and reconnaissance categories, restricting the generalizability of the findings to other IoT intrusion types. Additionally, although SMOTE effectively balances the dataset, reliance on synthetic oversampling raises concerns about potential overfitting and reduced robustness against unseen attack patterns. Another limitation is the high computational cost of the models, which can impede deployment in real-world IoT environments with resource constraints. Furthermore, the models were evaluated in a controlled dataset environment without real-world validation, reducing their practical deployment potential. Future work should address these challenges by incorporating diverse intrusion types, optimizing computational costs, and testing these models in real-world IoT network scenarios.

V. CONCLUSION

This study investigated deep learning techniques for intrusion detection using the BoT-IoT dataset, which exhibits a bias towards the DDoS/DoS category rather than achieving a balance across all categories. SMOTE was used to address this imbalance, which helped mitigate the overfitting issue for minority classes. Four deep learning models were developed for the BoT-IoT dataset, based on DNN, CNN, RNN, and LSTM, with each model achieving a different level of accuracy: 99.15%, 99.81%, 99.60%, and 99.97%, respectively. The evaluation results showed that LSTM struck a commendable balance between effectiveness and efficiency, outperforming state-of-the-art deep learning IDS approaches tested on the Bot-IoT dataset, achieving a multiclass classification accuracy of 99.97%. Future plans involve the evaluation of different classifiers across various datasets and real-world systems. Future research should also focus on determining the most effective deep learning methods for intrusion detection in IoT environments, focusing on improving accuracy, recall, training time, and reducing false alarm rates. Additionally, balancing the dataset through various oversampling techniques should be a key focus in future research efforts.

ACKNOWLEDGMENT

This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, Saudi Arabia (Grant No: IFPIP: 709-612-1443). The authors gratefully acknowledge the technical and financial support from the DSR.

REFERENCES

[1] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020, https://doi.org/10.1109/COMST.2020.2988293.

[2] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for IoT-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, Dec. 2018, Art. no. 21, https://doi.org/10.1186/s13677-018-0123-6.

[3] R. H. Altaie and H. K. Hoomod, "An Intrusion Detection System using a Hybrid Lightweight Deep Learning Algorithm," *Engineering, Technology & Applied Science Research*, vol. 14, no. 5, pp. 16740–16743, Oct. 2024, https://doi.org/10.48084/etasr.7657.

[4] S. M. S. Bukhari *et al.*, "Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with SCNN-Bi-LSTM for enhanced reliability," *Ad Hoc Networks*, vol. 155, Mar. 2024, Art. no. 103407, https://doi.org/10.1016/j.adhoc.2024.103407.

[5] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17–39, Oct. 2018, https://doi.org/10.1016/j.comnet.2018.07.017.

[6] C. Han, J. M. Jornet, E. Fadel, and I. F. Akyildiz, "A cross-layer communication module for the Internet of Things," *Computer Networks*, vol. 57, no. 3, pp. 622–633, Feb. 2013, https://doi.org/10.1016/j.comnet.2012.10.003.

[7] R. Alghamdi and M. Bellaiche, "An ensemble deep learning based IDS for IoT using Lambda architecture," *Cybersecurity*, vol. 6, no. 1, Mar. 2023, Art. no. 5, https://doi.org/10.1186/s42400-022-00133-w.

[8] A. Awajan, "A Novel Deep Learning-Based Intrusion Detection System for IoT Networks," *Computers*, vol. 12, no. 2, Feb. 2023, Art. no. 34, https://doi.org/10.3390/computers12020034.

[9] H. Alkahtani and T. H. H. Aldhyani, "Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms," *Complexity*, vol. 2021, no. 1, 2021, Art. no. 5579851, https://doi.org/10.1155/2021/5579851.

[10] M. A. Alsoufi *et al.*, "Anomaly-Based Intrusion Detection Systems in IoT Using Deep Learning: A Systematic Literature Review," *Applied Sciences*, vol. 11, no. 18, Jan. 2021, Art. no. 8383, https://doi.org/10.3390/app11188383.

[11] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions," *Electronics*, vol. 9, no. 7, Jul. 2020, Art. no. 1177, https://doi.org/10.3390/electronics9071177.

[12] A. Fatani, A. Dahou, M. A. A. Al-qaness, S. Lu, and M. Abd Elaziz, "Advanced Feature Extraction and Selection Approach Using Deep Learning and Aquila Optimizer for IoT Intrusion Detection System," *Sensors*, vol. 22, no. 1, Jan. 2022, Art. no. 140, https://doi.org/10.3390/s22010140.

[13] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep Learning-Based Intrusion Detection for IoT Networks," in *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Kyoto, Japan, Dec. 2019, pp. 256–25609, https://doi.org/10.1109/PRDC47002.2019.00056.

[14] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, Nov. 2019, https://doi.org/10.1016/j.future.2019.05.041.

[15] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques," in *Mobile Networks and Management*, vol. 235, J. Hu, I. Khalil, Z. Tari, and S. Wen, Eds. Springer International Publishing, 2018, pp. 30–44.

[16] S. M. Kasongo and Y. Sun, "A Deep Learning Method With Filter Based Feature Engineering for Wireless Intrusion Detection System," *IEEE Access*, vol. 7, pp. 38597–38607, 2019, https://doi.org/10.1109/ACCESS.2019.2905633.

[17] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, https://doi.org/10.1109/ACCESS.2019.2895334.

[18] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, "Toward a deep learning-based intrusion detection system for IoT against botnet attacks," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, Mar. 2021, Art. no. 110, https://doi.org/10.11591/ijai.v10.i1.pp110-120.

[19] "Power of Recurrent Neural Networks (RNN): Revolutionizing AI," *Simplilearn.com*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn.

[20] S. Saxena, "What is LSTM? Introduction to Long Short-Term Memory," *Analytics Vidhya*, Mar. 16, 2021. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

[21] P. Skryjomski and B. Krawczyk, "Influence of minority class instance types on SMOTE imbalanced data oversampling," in *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, Oct. 2017, pp. 7–21.

[22] A. Derhab, A. Aldweesh, A. Z. Emam, and F. A. Khan, "Intrusion Detection System for Internet of Things Based on Temporal Convolution Neural Network and Efficient Feature Engineering," *Wireless Communications and Mobile Computing*, vol. 2020, no. 1, 2020, Art. no. 6689134, https://doi.org/10.1155/2020/6689134.

[23] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, 2006.

[24] "Performance Metrics in Machine Learning." https://www.tutorialspoint.com/machine_learning/machine_learning_performance_metrics.htm.