Department: Head
Editor: Name, xxxx@email

Comparative Analysis of Enhanced Network Security in IoT Using Advanced Deep Neural Networks and Recurrent Neural Networks

Nafiz Mahamud
Master’s in Computer Science, The Catholic University of America

Dr. Minhee Jun
Assistant Professor, The Catholic University of America

***Abstract*— To enhance anomaly-based Intrusion Detection Systems (IDS), we developed and tested models using Deep Neural Networks (DNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Autoencoders (AE) on the IoT-23 dataset. Our approach achieved zero false negatives, improving detection capabilities against evolving security threats.**

■**THE INTRODUCTION** Telecommunication advancements are swiftly progressing, fostering greater global connectivity annually. Contemporary industries, like automotive transport, smart agriculture, and public safety, are embracing new standards and technologies such as Internet-of-Things (IoT) devices. An average IoT setup comprises three tiers: the perception layer, network layer, and application layer**.** The Internet of Things (IoT) is exposed to security risks across all its architectural levels and has been grappling with security issues since its inception [**1**]. Broadly, the perception layer faces threats such as malicious code injection, eavesdropping, and interference [**2**], [**3**]. Similarly, the network layer is vulnerable to spoofing, denial of service, man-in-the-middle, and routing information attacks [**3**]. Additionally, the application layer is susceptible to viruses, worms, and phishing attempts. A botnet represents a targeted form of cyber attack leveraging Internet of Things (IoT) devices. Angrishi et al. [**4**] define a botnet as a large collection of internet-connected devices manipulated to inundate a specified server (or servers) with simultaneous requests, rendering it incapable of responding to genuine requests, effectively halting its operation. This assault constitutes a distributed denial of service (DDoS) attack. These attacks have become increasingly sophisticated, making detection challenging [**4**]. IoT botnets pose a threat not only to the owners of IoT devices but also to all internet users. Because DDoS attacks require substantial network traffic to disrupt services, IoT devices are ideal hosts due to their vast numbers and generally weak security measures, rendering them easy targets [**5**]. In their research, Das et al. [**6**] illustrate Mirai, a recent example of such a botnet, wherein a virus seeks out susceptible devices and links them to Command-and-Control Servers (C&C servers). Das et al. [**6**] and Tushir et al. [**7**] observed that IoT devices linked to Mirai Botnets are primarily utilized to execute DDoS assaults on targeted devices. Tushir et al. [**8**] investigated the impact of Mirai attacks on IoT devices, revealing a 40% surge in energy consumption and a 50% increase in storage usage. Subsequently, numerous iterations and variants of the Mirai botnet emerged, such as Persirai, Hajime, and BrickerBot [**8**]. Targets of DDoS attacks encompassed websites, cloud providers, individuals, educational institutions, telecommunication companies, and DNS providers (Dyn), which serviced multiple websites including Reddit, Amazon, Spotify, Airbnb, among others [**5**].

As per Statista [**9**], the global count of IoT-connected devices reached nearly 8.74 billion in 2021, with a Cisco white paper [**10**] projecting a rise to approximately 30 billion by 2023, compared to roughly 18 billion in 2018. By 2024, it's estimated that there will be 83 billion devices connected to the Internet of Things (IoT) **[11]**. Moreover, the same paper **[10]** anticipates a surge in Distributed Denial of Service (DDoS) attacks to around 15 million by 2023, contrasting with 7 million recorded in 2018 [**12**]. According to statistics, the frequency of attacks is doubling annually, resulting in significant financial losses, amounting to tens of millions of dollars specifically from ransomware attacks [**13**].

One effective strategy for thwarting such assaults involves implementing a robust Intrusion Detection System (IDS) [**13**] capable of identifying various forms of intrusion. Presently, IDSs employ two main detection approaches: signature-based and anomaly-based. Signature-based detection systems are hindered in their effectiveness by their incapacity to recognize emerging cyber threats and their reliance on manual updates to the signature database which can be laborious. Conversely, anomaly-based methods analyze data, relying on the system's comprehension of typical behavior to flag any incoming connections that appear aberrant. Network intrusion detection seeks to assess diverse network data using different behavioral analyses to uphold its security. Numerous methods exist for detecting anomalies in networks. Although machine learning has proven indispensable and efficient in promptly identifying cyber-attacks, Deep learning using extensive datasets for training can potentially avoid overfitting issues, as it possesses greater capacity for generalization compared to conventional learning models [**12**]. To train an AI model effectively, high-quality, large-scale data from IoT devices are essential. However, IoT devices are vulnerable to cybersecurity threats. By combining IoT's real-time data collection with AI's data analysis and decision-making capabilities, organizations can develop more responsive, adaptive, and efficient systems across various sectors. Although several anomaly detection techniques are employed, there have been fewer comparative studies of different deep learning models for anomaly detection. Since intrusions entail a sequence of linked malevolent actions executed by an internal or external perpetrator to compromise the security of the designated system, our attention will be directed towards Recurrent Neural Networks, primarily tailored for sequential data processing.

Constructing an IDS for automatic cyber-attack detection necessitates a suitable dataset for training. We are going to explore the IoT-23 dataset by Garcia et al. [**14**] is a recent release specifically tailored to address cyber-attacks involving IoT devices, introduced in early 2020. Our proposed IDS use several deep learning-based models for the binary classification. There are several factors for selecting binary classification. First, binary classification simplifies the problem to distinguishing between normal (benign) and abnormal (malicious) activities. This can make the system easier to design, implement, and maintain.

Second, since the model only needs to learn two classes, training and prediction can be faster and require less computational resources. Finally, by focusing on identifying anomalies as a whole, the IDS can be more robust in catching new or unknown types of attacks that deviate significantly from normal behavior.

Literature Review

Li et al. [**17**] used convolutional neural networks (CNNs) for intrusion classification, dividing the dataset into four segments based on feature correlations. They transformed one-dimensional features into grayscale graphs and trained four CNNs (CNN1, CNN2, CNN3, CNN4) individually for binary classification. A combined model, CNN0, integrated these outputs and trained on the entire dataset. Using the NSL-KDD dataset, CNN1 achieved 82.62% and 67.22% accuracy on KDDTest+ and KDDTest-21, respectively. The ensemble model achieved 86.95% and 76.67% on the same test sets. Martin et al. [**21**] proposed a model that uses a linear classifier based on a Neural Network (NN) with linear activations. This model incorporates feature transformations using kernel approximation algorithms such as Nystrom, Random Fourier Features, and Fastfood transformation, which add the necessary complexity and non-linear characteristics to the model. To test their model, they chose three datasets but only performed binary classification on the NSL-KDD [18] dataset. The highest accuracy achieved in this binary classification was 80%. As evident, the results were not particularly promising. Kim et al. [**22**] developed a hybrid model integrating a convolutional neural network (CNN) and a long short-term memory network (LSTM) for the binary classification. They tested this model on two publicly available datasets, CSIC-2010 and CICIDS2017, achieving accuracies of 91.5% and 93.0%, respectively. Susilo et al. [**23**] utilized three algorithms—Random Forests, Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN)—to identify network intrusions. They employed the Bot-Iot dataset, created by UNSW Canberra [24] for multi-class classification. The CNN algorithm achieved the highest accuracy, reaching 91.25%. Yin et al. [**25**] introduced a deep learning method for intrusion detection using recurrent neural networks (RNN-IDS) and assessed its performance in both binary and multiclass classification tasks. They trained their model using the KDDTrain+ dataset and tested it with the KDDTest$^+$ and KDDTest^{-21} datasets, the latter being a subset of the former. By experimenting with different hyperparameters (such as the number of nodes and learning rate), they achieved the highest accuracy of 83.28% on the KDDTest$^+$ dataset and 68.55% on the KDDTest21 dataset, with the optimal configuration being 80 hidden nodes and a learning rate of 0.1. Sokolov et al. [**34**] explored the use of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), for intrusion detection for the binary classification in Industrial Control Systems (ICS), due to their ability to handle sequential data. The study emphasized the importance of considering both network traffic and the state of industrial processes for effective intrusion detection. The experiments compared the performance of LSTM and GRU networks in detecting intrusions using the Gas Pipeline dataset [35][36]. GRU networks showed slightly better performance with an accuracy of 91.70% compared to LSTM's 90.68%. The study found that GRU networks learn faster and are computationally more efficient than LSTMs. None of the papers didn't address dataset balancing or adequately report false rates. When one class significantly outnumbers another, models tend to become biased toward the majority class, leading to poor performance in detecting the minority class (anomalies). Thus, balancing the dataset helps the model generalize better to new, unseen data. High accuracy on an imbalanced dataset can be misleading. For example, if 95% of the data is normal and 5% is anomalous, a model predicting every instance as normal would achieve 95% accuracy but fail to detect intrusions. Balancing the dataset ensures that accuracy and other metrics truly reflect the model's performance and leads to better training dynamics, as gradient-based learning algorithms benefit from more stable gradients. Therefore, we balanced our dataset and included metrics such as False Positive and False Negative rates for comprehensive evaluation. Balancing false negatives and false positives is often the goal, but the operational context also influences prioritization. In stable environments, reducing false positives is crucial for operational efficiency. In dynamic environments, a low false negative rate is preferred to adapt to emerging threats. In high-security settings, minimizing false negatives is paramount, even at the cost of more false positives, to avoid missed detections leading to successful attacks. Hence, our primary aim is to minimize false negatives while maintaining an acceptable false positive rate.

Methodology

The Iot-23 Dataset

This study utilized the IoT-23 dataset, which comprises network traffic collected from Internet of Things (IoT) devices. It includes 20 instances of malware and 3 instances of benign activity. This dataset aims to provide a substantial collection of real-world labeled IoT malware infections and benign IoT traffic, facilitating the development of machine learning algorithms for researchers. It consists of 23 captures, referred to as scenarios, wherein 20 involve malicious activity and 3 involve benign activity. Each capture from infected devices may include the name of the potential malware sample executed in that scenario.

Here are the descriptions for each label in the IoT-23 dataset:

Attack: This denotes a form of assault originating from the infected device towards another host, exploiting vulnerabilities.
Benign: Indicates that no suspicious or malicious activities were detected within the connections.
C&C: Signifies that the infected device established a connection with a Command and Control (C&C) server.
DDoS: Indicates that the infected device is executing a Distributed Denial of Service attack.
FileDownload: Represents the downloading of a file to the infected device.
HeartBeat: Refers to packets sent through this connection to monitor the infected host by the C&C server.
Mirai: Implies connections displaying characteristics of a Mirai botnet.
Okiru: Implies connections displaying characteristics of an Okiru botnet.
PartOfAHorizontalPortScan: Denotes connections utilized for conducting a horizontal port scan to gather information for subsequent attacks.
Torii: Indicates connections demonstrating characteristics of a Torii botnet.

Furthermore, Zeek functions as software for conducting network analysis. The IoT-23 dataset utilized is in the conn.log.labeled format, derived from the original pcap file through Zeek network analyzer. Table 2 ([44], [45]) displays the dataset comprising 20 attributes.
Specific attributes, such as conn_state and history, hold significance with values or characters carrying particular implications. Refer to Tables 3 and 4 from [45] for descriptions of these attribute values.

Table 2: Features Description

	Feature	Description
1	ts	The time of the first packet
2	uid	A unique identifier of the connection
3	proto	The transport layer protocol of the connection
4	id.orig_h	Originator/Source IP Address
5	id.orig_p	Originator/Source Port number
6	id.resp_h	Responder/Destination IP Address
7	id.resp_p	Responder/Destination Port number
8	service	An identification of an application protocol
9	duration	How long the connection lasted
10	orig_bytes	The number of payload bytes the originator sent
11	resp_bytes	The number of payload bytes the responder sent
12	conn_state	The possible connection state values
13	local_orig	If the connection is originated locally, this will be T and F for remotely
14	local_resp	If the connection is responded locally, this will be T and F for remotely
15	missed_bytes	Indicate the number of bytes missed in content gaps
16	history	Records the state history of connections as a string
17	orig_pkts	Number of packets that the originator sent
18	orig_ip_bytes	Number of IP level bytes that the originator sent
19	resp_pkts	Number of packets that the responder sent
20	resp_ip_bytes	Number of IP level bytes that the responder sent

Table 3: Description of conn_state values

	State	Meaning
1	S0	Connection attempt seen, no reply
2	S1	Connection established, not

		terminated (0 byte counts)
3	SF	Normal establish & termination (>0 byte counts)
4	REJ	Connection attempt rejected
5	S2	Established, ORIG attempts close, no reply from RESP.
6	S3	Established, RESP attempts close, no reply from ORIG.
7	RSTO	Established, ORIG aborted (RST)
8	RSTR	Established, RESP aborted (RST)
9	RSTOS0	ORIG sent SYN then RST; no RESP SYN-ACK
10	RSTRH	RESP sent SYN-ACK then RST; no ORIG SYN
11	SH	ORIG sent SYN then FIN; no RESP SYN-ACK (“half open”)
12	SHR	RESP sent SYN-ACK then FIN; no ORIG SYN
13	OTH	No SYN, not closed. Midstream traffic. Partial

Table 4: Description of history values

State	Meaning
S	a SYN without the ACK bit set
H	a SYN & ACK (“handshake”)
A	a pure ACK
D	packet with payload (“data”)
F	packet with FIN bit set
R	packet with RST bit set
C	packet with a bad checksum
I	Inconsistent packet (Both SYN & RST)

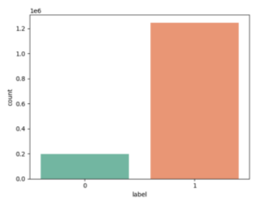
We encountered a minimal number of missing values, which we addressed by substituting them with zeros. Features `local_orig', `local_resp' which were empty for all the files (scenarios) and hence they were dropped. Based on previous work on pre-processing of intrusion detection datasets like NSL-KDD, CICIDS2017, features id.orig_h, id.orig_p, id.resp_h and id.resp_p contained IP adresses and port numbers were dropped. Additionally, the 'history' attribute, representing a sequence detailing the connection history, was initially dropped.

Since our focus was on binary classification, we assigned a value of '1' for all attack instances and '0' for benign instances in the "label" column using Python's 'map' function. Following the encoding of object type features, we examined each column and observed that the majority of values in certain features were zero. Consequently, we decided to drop those features from consideration as well [Table 5].

Table 5: Additional Dropped features after encoding

Column	Unique Values	Value Counts
Conn_state_RSTO	0	1444521
	1	153
Conn_state_RSTOS0	0	1444644
	1	30
Conn_state_RSTR	0	1444123
	1	551
Conn_state_S2	0	1444647
	1	27
Conn_state_S3	0	1444217
	1	2457

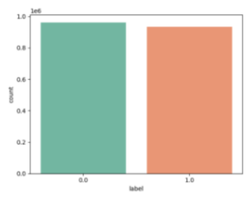
After removing all those columns, we now have a total of 15 columns, including the label column.



1 1246865

0 197809

Imbalanced



0.0 961750

1.0 934971

Balanced

Figure 1. Dataset Balancing

Initially, we divided the dataset into two segments: training set (75%) and test set (25%). Subsequently, we constructed a sequential deep neural network (DNN) comprising four layers, culminating in an output layer with a single node for binary classification. Training the network on the training set yielded an accuracy of 89.3%.

Our objective now is to generate some randomly distributed normal data. We first constructed a dataset comprising only benign/normal flows, identified by a label column value of '0'. Next, we calculated the standard deviations of all 14 columns using the 'std()' function. We then scaled down each standard deviation by a factor of 0.1 to reduce and normalize the deviation values, converting them into a 'numpy' array. A function named 'random_val()' was created to generate random values using a normal distribution via the 'tf.random.normal' function, with parameters including a seed value range of 1 to 256, a shape of 148534 rows and 14 columns, a mean of 0, and standard deviations derived from each column of the normal data. Finally, we employed our trained model to predict benign data from the generated random values. By iterating through a loop, we obtained the desired quantity of benign data and appended it to the original benign data. Following this procedure, we eliminated the duplicated entries and ultimately obtained a dataset that is reasonably balanced, as depicted in Figure 1(Right).

DNN: Utilizing the Rectified Linear Unit (ReLU) activation function across all hidden layers, our model is optimized using the "adam" optimizer, a widely used and optimized gradient descent algorithm, with the "binary_crossentropy" serving as the loss function.

Table 6. Arch. of DNN

Layer (Type)	Output Shape	Number of parameters
Input(Dense)	(None, 100)	1500
Dense	(None, 50)	5050
Dense	(None, 40)	2040
Dense	(None, 40)	1640
Output(Dense)	(None, 1)	41
Total parameters: 10,271 Trainable parameters: 10,271 Non-trainable parameters: 0		

LSTM: We developed a very light-weighted sequential LSTM model comprising just three hidden (LSTM) layers with limited nodes. In the final hidden layer, we allocated only five nodes and excluded sequence return, opting instead for the dense layer as the output layer (Output at Final Time Step).

Table 7. Arch. of LSTM

Layer (Type)	Output Shape	Number of parameters
Input(LSTM)	(None, None, 20)	1760
LSTM	(None, None, 10)	1240
LSTM	(None, 5)	320
Output(Dense)	(None, 1)	6
Total parameters: 3326 Trainable parameters: 3326 Non-trainable parameters: 0		

Gated Recurrent Unit (GRU): To improve outcomes, we adopted a GRU model featuring three layers of hidden units. In the initial layer, we incorporated a one-dimensional Convolutional layer to compress the data, employing a filter count of five, a kernel size of two, a stride of two, and valid padding. For the second layer, we utilized just five nodes and ensured sequence retention. In the last hidden layer, we employed only three nodes without sequence retention.

Table 8. Arch. of GRU

Layer (Type)	Output Shape	Number of parameters
Input(Conv1D)	(None, None, 5)	15
GRU	(None, None, 5)	180
GRU	(None, 3)	90
Output(Dense)	(None, 1)	4
Total parameters: 289 Trainable parameters: 289 Non-trainable parameters: 0		

LSTM-Autoencoder: In our stacked auto-encoding setup, we utilized LSTM layers as the hidden layers due to their effectiveness in sequence learning. During the encoding phase, the model compressed the data from ten dimensions down to three (with three representing the latent representation). In the decoding phase, two hidden layers were employed to restore the dimensions from three back to ten.

Table 9. Arch. of Autoencoder(LSTM)

Layer (Type)	Output Shape	Number of parameters
Input(LSTM)	(None, None, 10)	
LSTM	(None, None, 5)	
LSTM	(None, None, 3)	
RepeatVector	(None, 1)	
LSTM	(None, None, 5)	800
LSTM	(None, None, 10)	966
Output(Dense)	(None, 1)	
Total parameters: 1,766 Trainable parameters: 1,766 Non-trainable parameters: 0		

GRU-Autoencoder: Likewise, we employed the GRU layer for autoencoding purposes.

Table 10: Arch. of Autoencoder(GRU)

Layer (Type)	Output Shape	Number of parameters
Input(GRU)	(None, None, 10)	
GRU	(None, None, 5)	
GRU	(None, None, 3)	
RepeatVector	(None, 1)	
GRU	(None, None, 5)	800
GRU	(None, None, 10)	966
Output(Dense)	(None, 1)	
Total parameters: 1,766 Trainable parameters: 1,766 Non-trainable parameters: 0		

Bi-directional RNN: A Bi-RNN consists of two separate RNNs: one that processes the sequence forward (from the start to the end) and another that processes it backward (from the end to the start).In the initial hidden layer, we utilized a basic RNN layer, while in the subsequent hidden layer, we employed an LSTM layer for bidirectional processing to address the previously mentioned challenges associated with RNNs.

Table 11. Arch. of Bi-RNN

Layer (Type)	Output Shape	Number of parameters
Input (RNN)	(None, None, 20)	440
Bidirectional (LSTM)	(None, None, 20)	2480
Output(Dense)	(None, 1)	21
Total parameters: 2,941 Trainable parameters: 2,941 Non-trainable parameters: 0		

Machine Learning & Deep Learning: We developed an architecture where a Deep Neural Network (DNN) was used for feature extraction, and a Random Forest (RF) was employed for binary classification. The DNN was chosen to capture and represent the complexities and non-linearities in the dataset, which enabled the RF classifier to achieve more accurate intrusion detection. Our hypothesis proved correct, as this approach resulted in better accuracy compared to several other models, specifically in reducing both False Positives and False Negatives.

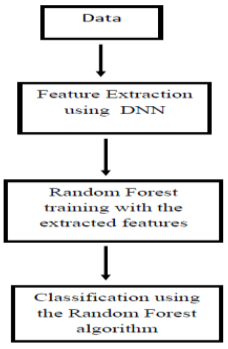

Data
Feature Extraction using DNN
Random Forest training with the extracted features
Classification using the Random Forest algorithm

Figure 2. ML & DNN Arch.

To implement this, we first divided our dataset into two parts: 75% for training and 25% for testing. The training set was further split into 60% for actual training and 40% for validation. The DNN was trained using the 60% training data and validated with the remaining 40%. The DNN outputted extracted features as ten-dimensional vectors. These features, along with their corresponding labels, were then used to train the RF classifier. Finally, the classifier was tested on the 25% testing data, leading to superior results.

Table 12: Arch. of DNN

Layer (Type)	Output Shape	Number of parameters
Input(Dense)	(None, 100)	1500
Dense	(None, 50)	5050
Dense	(None, 40)	2040
Dense	(None, 10)	410
Total parameters: 9000 Trainable parameters: 9000 Non-trainable parameters: 0		

Experiments

In this study, we used Keras on the backend Tensorflow (Version: 2.10.0), known for its speed, simplicity and popularity in deep learning. The experiment was conducted using Jupyter Notebook (Version: 6.4.12) on a HP Pavilion personal computer equipped with an Intel Core i7-1195G7 CPU @ 2.90GHz and 16 GB of RAM.

Table 18. Accuracy of the models

Model	Accuracy
DNN	94.41
LSTM	96.10
GRU	96.23
LSTM & AE	94.39
GRU & AE	94.44
Bi-Directional RNN	94.47
ML & DNN	96.21

Table 14. Precision, Recall, & F1

Model (Thr>0.5)	Label	Precision	Recall	F1
DNN	0	1.00	0.89	0.94
	1	0.90	1.00	0.95
LSTM	0	0.99	0.93	0.96
	1	0.93	0.99	0.96
GRU	0	1.00	0.93	0.96
	1	0.93	1.00	0.96
LSTM & AE	0	1.00	0.89	0.94
	1	0.90	1.00	0.95
GRU & AE	0	1.00	0.89	0.94
	1	0.90	1.00	0.95
Bi-Directional RNN	0	1.00	0.89	0.94
	1	0.90	1.00	0.95
ML & DNN	0	0.99	0.93	0.96
	1	0.93	0.99	0.96

Table 15. False Positive & False Negative Rates

Model	F/P	F/N
DNN	10.9	0.2
LSTM	7.2	0.5
GRU	7.0	0.5
LSTM & AE	11.0	0.0
GRU & AE	10.9	0.1
Bi-Directional RNN	10.8	0.1
ML & DNN	6.9	0.5

Table 16. Comparison of binary classification models using deep learning for anomaly detection.

Article	Model	Year	Accuracy %(max.)
Martin et al.[21]	NN	2019	80.00
Li et al. [17]	CNN	2020	86.95
Susilo et al. [23]	CNN	2020	91.25
Sokolov et al. [34]	GRU	2019	91.70
Our Proposed	ML+DNN	2024	96.21
Our Proposed	GRU	2024	96.23

Discussion

Through multiple experiments, we have demonstrated the effectiveness of the proposed models in addressing not only reducing the model's features number but also enhancing its classification detection ability. Despite the notable improvements in detection accuracy attained by the proposed models, there remain certain limitations that warrant attention. In particular, the rate of false positives. Future research endeavors may include the exploration of alternative strategies to enhance the binary classification performance of the proposed models. This may involve investigating diverse model architectures, such as incorporating attention mechanisms or exploring novel loss functions that are better equipped to capture the unique characteristics of the dataset or generate more data to extract more insightful information. Additionally, efforts could be made to improve the interpretability of the model by analyzing the attention weights of the models and identifying significant features for intrusion detection. These approaches may culminate in further improvements in the overall detection performance and can have far-reaching implications beyond the scope of intrusion detection. Overall, this study contributes to the knowledge system by proposing several with novel approaches and demonstrating its effectiveness in addressing those issues in intrusion detection models, while also emphasizing the need for further research and improvement in this area.

Conclusion

The rapid rise in the use of IoT devices has turned them into unsuspecting vectors for cyber-attacks. This paper demonstrates that for certain attacks and IoT devices, deep learning methods such as RNN and BiRNN can effectively classify attacks with high accuracy. While there are numerous datasets available for intrusion detection, it is preferable to use a dataset specifically generated from IoT devices. Thus, this study utilizes the recent IoT-23 dataset. We implemented baseline models, including GRU+AE and DNN+ML. Our findings indicate that RNN-based models are particularly effective (0% False Negative) in identifying and classifying attacks. Additionally, we have shown the effective use of DNN for feature extraction and ML for classification. Given the critical role of AE, future research could explore anomaly detection further using various types of AE, such as Sparse, Generative AE and Variational AE.

■REFERENCES

1. C. Vorakulpipat, E. Rattanalerdnusorn, P. Thaenkaew, and H. D. Hai, “Recent challenges, trends, and concerns related to IoT security: An evolutionary study,” in Proc. 20th Int. Conf. Adv. Commun. Technol.(ICACT), Feb. 2018, pp. 405_410, doi: 10.23919/ICACT.2018.8323774.
2. S. Fenanir, F. Semchedine, S. Harous, and A. Baadache, “A semisupervised deep auto-encoder based intrusion detection for IoT,” Ingénierie Syst. Inf., vol. 25, no. 5, pp. 569_577, Nov. 2020, doi:10.18280/isi.250503.
3. M. Burhan, R. A. Rehman, B. Khan, and B.-S. Kim, “IoT elements, layered architectures and security issues: A comprehensive survey,” Sensors, vol. 18, no. 9, Sep. 2018, Art. no. 9, doi: 10.3390/s18092796.
4. K. Angrishi, “Turning Internet of Things (IoT) into internet of vulnerabilities (IoV): IoT botnets,” 2017, arXiv:1702.03681.
5. R. Ahmad and I. Alsmadi, “Machine learning approaches to IoT security: A systematic literature review,” Internet Things, vol. 14, Jun. 2021, Art. no. 100365, doi: 10.1016/j.iot.2021.100365.
6. S. Das, P. P. Amritha, and K. Praveen, “Detection and prevention of mirai attack,” in Proc. Soft Comput. Signal Process., Singapore, 2021, pp. 79-88.
7. B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, “The impact of DoS attacks on resource-constrained IoT devices: A study on the Mirai attack,” 2021, arXiv:2104.09041.
8. C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “DDoS in the IoT: Mirai and other Botnets,” Computer, vol. 50, no. 7, pp. 80_84, 2017, doi: 10.1109/MC.2017.201.
9. IoT connected devices worldwide 2019-2030. Accessed: Jun. 17, 2021. [Online]. Available: https://www.statista.com/statistics/1183457/iotconnected-devices-worldwide/
10. Cisco Annual Internet Report_Cisco Annual Internet Report (2018-2023) White Paper. Accessed: May 31, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executiveperspectives/annual-internet-report/white-paper-c11-741490.html
11. S. Smith. (2020). IoT Connections To Reach 83 Billion By 2024, Driven By Maturing Industrial Use Cases. Accessed: Apr. 10, 2021. [Online]. Available:https://www.juniperresearch.com/press/press-releases/iot-connections-to-reach-83-billion-by-2024-driven
12. S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, “Intrusion detection systems in the Internet of Things: A comprehensive investigation,” Comput. Netw., vol. 160, pp. 165-191,Sep.2019, doi:10.1016/j.comnet.2019.05.014.
13. K. A. Jallad, M. Aljnidi and M. S. Desouki, “Anomaly detection optimization using big data and deep learning to reduce false-positive,” J Big Data, Vol. 7, Aug. 2020, Art. No. 68 (2020), doi: 10.1186/s40537-020-00346-1.
14. S. Garcia, A. Parmisano, and M. J. Erquiaga, “IoT-23: A labeled dataset with malicious and benign IoT network traf_c (version 1.0.0),” Zenodo, vol. 20, p. 15, Jan. 2020, doi: 10.5281/zenodo.4743746.
15. Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, "Robust detection for network intrusion of industrial IoT based on multi-CNN fusion," Measurement, vol. 154, Mar. 2020, Art. no. 107450, doi:10.1016/j.measurement.2019.107450.
16. M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," Expert Syst. Appl., vol. 124, pp. 196-208, Jun. 2019, doi: 10.1016/j.eswa.2019.01.063.
17. A. Kim, M. Park and D. H. Lee, "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," in IEEE Access, vol. 8, pp. 70245-70261, 2020, doi: 10.1109/ACCESS.2020.2986882.
18. B. Susilo and R. F. Sari, "Intrusion detection in IoT networks using deep learning algorithm,"

Information, vol. 11, no. 5, p. 279, May 2020, doi:10.3390/info11050279.

19. 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 Gener. Comput. Syst., Vol. 100, pp.779–796, Nov. 2019. doi: 10.1016/j.future.2019.05.041.

20. C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," in IEEE Access, vol. 5, pp. 21954-21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

21. A. N. Sokolov, S. K. Alabugin and I. A. Pyatnitsky, "Traffic Modeling by Recurrent Neural Networks for Intrusion Detection in Industrial Control Systems," 2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russia, 2019, pp. 1-5, doi: 10.1109/ICIEAM.2019.8742961.