Journal of Information Systems Engineering and Management
2025, 10(32s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

CLAE-MLP: A Deep Learning Framework for Botnet Detection in IoT Network Using N-BaIoT Dataset

Rituparna Borah[1], Satyajit Sarmah[2*] , Manas Pratim Kalita[3], Ranjan Patowary[4]
[1]Department of Information Technology, Gauhati University. Email: r264porna@gmail.com
[2*]Department of Information Technology, Gauhati University. Email: ss@gauhati.ac.in
[3]Department of Information Technology, Gauhati University. Email: manaspratimkalita4@gmail.com
[4]Central Institute of Technology, Kokrajhar. Email: r.patowary@cit.ac.in

Citation: Satyajit Sarmah, et al. (2025), CLAE-MLP: A Deep Learning Framework for Botnet Detection in IoT Network Using N-BaIoT Dataset, *Journal of Information Systems Engineering and Management*, *10*(30s), xyz,

ARTICLE INFO

Received: 29 Dec 2024

Revised: 15 Feb 2025

Accepted: 24 Feb 2025

ABSTRACT

Modern industries and day-to-day activities have experienced major progress from the Internet of Things because it links devices instantly to share real-time data. Internet of Things devices connecting to each other creates exposure to cyber-attacks and produces botnet attacks that damage network security while spreading data breaches and blocking service access. To protect IoT networks you must detect and stop botnet threats. This research creates a deep learning model that combines CNN autoencoders with LSTMs and MLPs to better find botnet attacks in IoT networks. The CNN module generates spatial understanding of data patterns while the LSTM module finds temporal sequences. The MLP module helps make better predictions and reduces incorrect findings. Our system can check network traffic quickly to find botnets that protect digital security. Experimental tests prove that this model finds both familiar and new security threats to strengthen IoT protection.

Keywords: IoT Security, Botnet Detection, Deep Learning, CNN-LSTM- MLP, Cybersecurity Threats

1. Introduction

Online services together with the internet have evolved into crucial elements which people use daily [1]. The IoT expansion powers this advancement which has revolutionized work practices and communication systems and connectivity methods. [2]. A network of connected devices creates IoT through data-sharing capabilities over the internet. [3-4]. The transformative power of IoT exists because of its ability to share data in real time which affects both business operations and everyday life. [5]. The global IoT device installation count reached 35.82 billion during 2021 before experts expect it to surpass 75.44 billion by 2025 and exceed 25.4 billion active devices by 2030. [6] The quick expansion of IoT demonstrates its influence on multiple business areas and market segments. [7] Cyber attackers use IoT's interconnected structure as a means to commit harmful actions while enjoying its beneficial aspects. Botnets have become a common tool for launching cyberattacks on systems enabled by IoT devices which requires immediate detection for cybersecurity protection. Network security and user trust depend on botnet identification followed by their breakdown to stop DDoS attacks and data breaches and malware distribution. When security threats get detected early the damage stays under control and operational effectiveness increases while cyber defenses become strengthened. Deep learning methods strengthen botnet detection through effective recognition of security threats with quick threat identification mechanisms which shield infrastructure from financial losses and operational risks. operational risks. Deep learning emerges as an effective tool to combat botnet attacks because of its increasing importance against growing cyber threats. The detection capabilities of botnets get significantly improved through deep learning because it enables automatic intelligent threat analysis. The system implements deep neural networks that successfully detects current botnets and those that emerge in the future. Real-time identification of malignant actions becomes possible through extracting features from network packets at their headers [8]. The technology of deep learning is undergoing modifications to process encrypted data while also detecting new botnet architecture patterns. Scientists continue to work on research that improves detection models to handle changing botnet patterns and to increase their ability to detect various patterns [9]. The authors create a deep learning framework which unites CNN autoencoders with LSTM and MLP to detect IoT botnets. The spatial feature extraction of CNN pairs effectively with LSTM's advanced temporal pattern recognition ability to produce enhanced classification performance through MLP thus establishing improved IoT security standards. The paper follows this organization: Section 2 reviews literature alongside existing research and technologies. Section 3 describes the proposed methodology. The fourth section shows the obtained results while analyzing the consequences of the proposed method implementation. Finally, Section 5 concludes the research.

2. Literature Review

In [10], GWO is applied to improve Intrusion Detection Systems (IDS) by feature selection. GWO was integrated with Extreme Learning Machine (ELM) for the optimized classification and was tested on UNSW-NB15 dataset. Concerned with generic attack detection, their approach brought the crossover error under 30.
The authors in [11] applied a combined CNN-LSTM model to detect Distributed Denial of Service (DDoS) attacks using CICIDS 2017 dataset. The accuracy of their model was 97.16.
In [12] the authors combine CNN and LSTM for botnet detection and achieve strong performance. DNNBoT1 had a training accuracy of 90.71.
The author in [13] examined the cybersecurity threats in autonomous vehicles (AVs) and proposed a deep learning model with CNNs and LSTMs for threat detection. The model achieves high accuracy and low false positives on real time vehicle data, and is practically applicable to AV cybersecurity.
In [14], the researcher used a Long Short Term Memory Autoencoder (LAE) to compressed large scale IoT network traffic while keeping key features.
In [15], the authors presented a hybrid deep learning model that uses CNNs for feature extraction and LSTMs for temporal pattern discovery in IoT botnet detection. The model was able to achieve high accuracy, precision and recall in identifying sophisticated attacks and improving IoT cybersecurity.

3. Proposed Methodology

This section offers an in-depth analysis of the botnet dataset used, deep learning models used and the proposed CLAE-MLP model.

3.1 Dataset Description

The N-BaIoT dataset, sourced from a machine learning repository, focuses on IoT network traffic for botnet detection. It consists of 155 features collected through switch port mirroring from nine commercial IoT devices. The dataset captures real network traffic, including botnet attacks such as Mirai and Bashlite. Table 1 presents the types and names of IoT devices used, while Table 2 outlines the 23 primary features recorded at intervals of 100ms, 500ms, 10s, 1min, and 10min.
Figure 1 shows the lab setup for collecting IoT botnet attacks. Devices connected via Wi-Fi access points, with port mirroring on switches capturing network traffic. Wireshark recorded the activity. Table 3 outlines BASHLITE and Mirai attacks—BASHLITE, a C-based DDoS attack, targets IoT cameras, while Mirai, discovered in 2016, exploits ARC processors in large-scale IoT networks.

Table1: Types and names of the devices

Device type	Device name
Dorbell	Daminin
	Emnio
Thermostat	Ecobee
Baby Monitor	Philips B120 N/10
Security Camera	Provision PT-737E
	Provision PT-838
	Simple Home XCS7-1002-WHT
	Simple Home XCS7-1003-WHT
Webcam	Samsung SNH101N

Table 2: Features

Aggregated by	Value	Statistic	Total No. of features
Source IP	Packet size(only outbound)	Mean, Variance	
	Packet Count	Integer	3
Source MAC_IP	Packet size (only outbound)	Mean, Variance	3
	Packet Count	Integer	
Channel	Packet size(only outbound)	Mean, Variance	
	Packet Count	Integer	10
	Amount of time between packer arrivalas	Mean,Variance,Integer	
	Packet size (both inbound and outbound)	Magnitude radius,covariance,correlation,coefficient	
Socket	Packet size(only outbound)	Mean,Variance	
	Packet count	Integer	7
	Packet size(both inbound and outbound)		
	Total		23

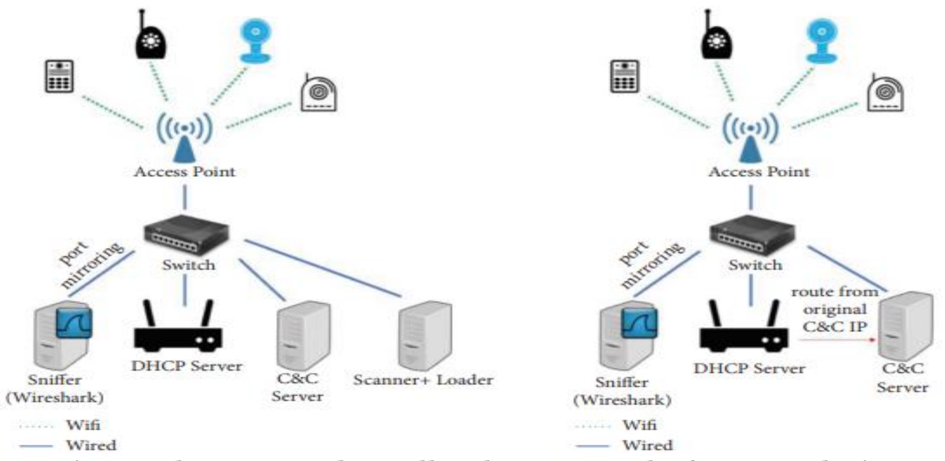

Fig 1: Lab setup used to collect botnet attacks from IoT devices

Table: Outline of attack patterns

Major attacks	Subattacks	Description
Bashlite	Junk	By sending spam data
	TCP Flood	Sends flood of request
	UDP Flood	Sends flood of request
	Scan	Scan the network for victim devices
	COMBO	Opens connection IP address and network port by sending spam data
Mirai	ACK	Sends flood of acknowledgement
	SYN	Sends synchronize- packet-flood
	Plain UDP	UDP flood by optimizing seeding packet per second Scans the network for victim devices
	Scan	

3.2 Deep Learning Models

The proposed hybrid deep learning model is based on the following three deep learning models i.e. Autoencoder, CNN-LSTM and MLP. The hyperparameters used for CNN-LSTM and MLP is shown in Table 4 and Table 5.

3.2.1 Autoencoders

Autoencoders are neural networks tailored for performing unsupervised learning. Autoencoders are made up of an encoder and a decoder, aiming to compress and then reconstruct the input data. This approach can be useful for tasks such as dimensionality reduction, feature extraction, or data reconstruction.

Encoder function: Let's represent the encoder as the function f(x), where x is the input data. The encoder transforms the input data into a lower-dimensional latent space (or space vector) representation.

$z=f(x)$ (1)

Decoder Function: The decoder function is represented as g(z), where z is the compressed latent representation. The decoder reconstructs the compressed representation, transforming it back to the original input space.

$(\hat{x}=g(z))$ (2)

Objective Function: One of the objectives of an autoencoder is typically to minimize the reconstruction error, which is the difference between the input data x and the output generated by the decoder $\hat{x}$. This error can also be used to detect anomalies or attacks. The objective function, often referred to as the loss function, can be expressed as:

$(L(x, \hat{x})=||x-\hat{x}||^2)$ (3)

where ‖·‖ represents an appropriate norm, like the Euclidean norm.

Table 4: Hyperparameters of CNN-LSTM

Hyperparameters	Value
Convolution Filters	192
Kernel size of filter	6
Fully connected layers	4
Activation function	Relu
Classification function	Sigmoid
Optimizer	Adam
Epochs	100

Table 5: Hyperparameters of MLP model

Hyperparameters	**Value**
Dense layers size	256,128,64
Activation function	Relu for hidden layers (Softmax for output layer
Dropout rate	0.3

3.2.2 CNN-LSTM networks

A CNN-LSTM network combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to process sequential data with spatial features. The CNN extracts spatial patterns, while the LSTM captures temporal dependencies. In this architecture, CNN layers generate feature maps that retain spatial hierarchies, which are then processed by LSTM layers to model sequential patterns.

- CNN Feature Extraction:

The input sequence (e.g., a frame from a video or a segment of a time series) is processed by the CNN layers to extract spatial features:

$f_t = \mathrm{CNN}(x_t)$

where x_t is the input at time step t and f_t is the feature map output from the CNN.

- LSTM Unit Representation:

The extracted features f_t are then fed into the LSTM unit, which can be represented as follows:

- Input Gate(i):

$i_t=\sigma(W_{ii}f_t+b_{ii}+W_{hi}h_{t-1}+b_{hf})$ (5)

- Forget Gate(f):

$f_t= \sigma(W_{if}f_t+b_{if}+W_{hf}h_{t-1}+b_{hf})$ (6)

- Cell Gate(g)

$g_t= \tanh(W_{ig}f_t+b_{ig}+W_{hg}h_{t-1}+b_{hg})$ (7)

- Output Gate (o)

$o_t= \sigma(W_{io}x_t+b_{io}+W_{ho}h_t+b_{ho})$ (8)

- Cell Gate (g)

$C_t=f_tC_{t-1}+i_tg_t$ (9)

- Hidden State(h)

$h_t=o_t\tanh(C_t)$ (10)

3.2.3 MLP (Multilayer Perceptron)

A Multilayer Perceptron (MLP) is an artificial neural network with multiple layers that model complex data relationships. Used for classification and regression, it consists of an input layer, hidden layers, and an output layer. The input layer represents dataset features and passes data to the hidden layers. Each hidden layer neuron applies a weighted sum, bias, and activation function. The output layer generates final predictions.

3.3 Proposed CLAE-MLP model

The CLAE-MLP (CNN-LSTM-Autoencoder-Multilayer Perceptron) model is a hybrid framework combining CNN-LSTM for sequence analysis and feature extraction with an MLP decoder for classification. The CNN-LSTM Autoencoder efficiently captures spatial and temporal dependencies, encoding crucial sequential patterns in the data. As shown in Figure 2, the model consists of a CNN-LSTM encoder and an MLP decoder, leveraging deep hierarchical feature learning to enhance IoT botnet detection. Integrating CNN, LSTM, and MLP strengthens anomaly detection by combining spatial feature extraction, sequential pattern recognition, and non-linear classification. This hybrid approach improves IoT intrusion detection by efficiently managing data dimensionality and boosting detection accuracy, ensuring robust security against evolving cyber threats.

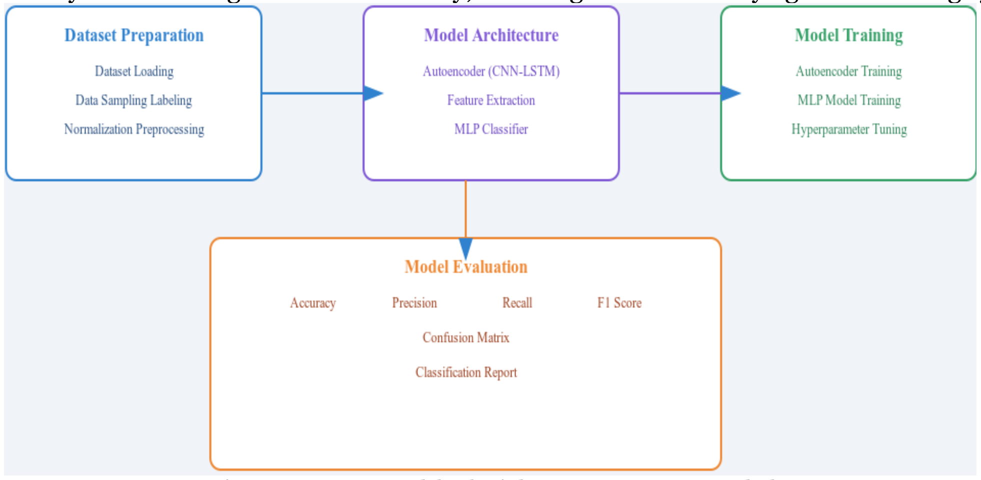

Fig. 2: Proposed hybrid CLAE-MLP Model

CLAE-MLP Algorithm
Input:
- Training dataset with labeled instances (x_{train},y_{train})
- Testing dataset for evaluation (x_{test})

Output:
- Anomaly predictions for the testing dataset.

Step 1: Initialize CNN-LSTM Autoencoder (CLAE) and MLP with specified architecture
Step 2: Training the CNN-LSTM:
1. For each epoch t and sequence (x_i,y_i):
- Forward pass-through CNN: $f^{(t)}$=CNN($X^{(t)}$)
- Forward pass-through LSTM: $h^{(t)}$=LSTM($f^{(t)}$)
- Compute loss: $\mathcal{L}_{CLSTM}$=Loss $(y, \hat{y})$
- Backpropagation to compute gradients: $\frac{\partial \mathcal{L}_{CLSTM}}{\partial W_{CLSTM}}, \frac{\partial \mathcal{L}_{CLSTM}}{\partial b_{CLSTM}}$
- Update CNN-LSTM weights:

$$W_{CLSTM} \leftarrow W_{CLSTM} - \alpha_{CLSTM} \frac{\partial \mathcal{L}_{CLSTM}}{\partial W_{CLSTM}}$$
$$b_{CLSTM} \leftarrow b_{CLSTM} - \alpha_{CLSTM} \frac{\partial \mathcal{L}_{CLSTM}}{\partial b_{CLSTM}}$$

Step 3: Encode Sequences with Autoencoder (CLAE)
1. For each sequence x_i in x_{train}
- Forward pass through CNN: $f^{(t)}$=CNN($X^{(t)}$)
- Forward pass through LSTM: $h^{(t)}$=LSTM($f^{(t)}$)

Step 4: Train MLP
- Feed sequence into Autoencoder (CLAE): z_{CLAE}=CLAE($h^{(t)}$)
- Forward pass through MLP: z^{l+1}=σ($W^{(t)}z^{(l)}$+$b^{(l)}$)
- Compute loss: $\mathcal{L}_{MLP=}$ Loss $(y, \hat{y})$
- Backpropagation to compute gradients: $\frac{\partial \mathcal{L}_{MLP}}{\partial W^t}, \frac{\partial \mathcal{L}_{MLP}}{\partial b^t}$
- Update MLP weights:

$$W^{(t)} \leftarrow W^{(t)} - \alpha_{MLP} \frac{\partial \mathcal{L}_{MLP}}{\partial W^t}$$

$b^{(l)} \leftarrow b^{(l)} - \alpha \frac{\partial \mathcal{L}_{MLP}}{\partial b^{(l)}}$

Step 5: Inference Phase:
1. For each instance x_i in x_{test}
- Forward pass through CNN: $f^{(t)}$=CNN($X^{(t)}$)
- Forward pass through LSTM: $h^{(t)}$=LSTM($f^{(t)}$)
- Encode the sequence with Autoencoder (CLAE): z_{CLAE}=CLAE($h^{(t)}$)
- Forward pass through MLP:z^{l+1}= σ($W^{(t)}z^{(l)}$+$b^{(l)}$)

Initializing and Training CNN-LSTM: The process begins with initializing a CNN-LSTM network, including setting of hyperparameters like the number of epochs, layers, units, batch size, and learning rate. The CNN extracts spatial features from the input data, which are then passed to the LSTM to capture temporal dependencies. For each epoch and sequence (x_i,y_i) a forward pass through the CNN-LSTM generates hidden states $h^{(t)}$. The loss $\mathcal{L}_{CLSTM}$ is computed by comparing the predicted output $\hat{y}$ with the true label y. Gradients are calculated via backpropagation, and the CNN-LSTM weights (W_{CLSTM}) and biases (b_{CLSTM}) are updated accordingly.

Encode Sequences with Autoencoder (CLAE): For each sequence x_i in the training set, the process begins with a forward pass through the CNN, which extracts spatial features $f^{(t)}$ from the input $x^{(t)}$. These features are then passed through the LSTM to capture temporal patterns, resulting in hidden states $h^{(t)}$. Finally, the hidden states $h^{(t)}$ is fed into the Autoencoder (CLAE) to encode the sequence into a lower-dimensional representation z_{CLAE} , which preserves the essential features of the sequence for further processing.

Train MLP: For each epoch t and instance (x_i,y_i), the MLP performs a forward pass where the input $z^{(l)}$ is processed through the network using weights $W^{(t)}$ and biases $b^{(l)}$, applying the activation function σ to produce the output $z^{(l+1)}$. The loss $\mathcal{L}_{MLP}$ is calculated by comparing the predicted output $\hat{y}$ with the actual label y. Backpropagation is then used to compute the gradients of the loss with respect to the weights and biases. Finally, the MLP weights and biases are updated using these gradients to improve the model's accuracy.

Inference Phase:
For each test instance x_i, the model first processes the input through the CNN to extract spatial features $f^{(t)}$. These features are then passed through the LSTM to capture temporal dependencies, resulting in hidden states $h^{(t)}$. The hidden states are encoded using the Autoencoder (CLAE) into a compact representation z_{CLAE}. Finally, this encoded sequence is passed through the MLP, where it undergoes a forward pass using the weights $W^{(t)}$ and biases $b^{(l)}$ to produce the final prediction z^{l+1}.

4. Results and Discussion

In this section, we present a brief summary of the results obtained.

4.1 Environment Setup

The proposed system was developed utilizing the hardware and software environment outlined in Table 6.

Table 6: Hardware and Software Environment

Hardware/Software	Requirement
Operating System	Windows 10
CPU	2.40 GHz
Memory	8GB
Development environment	Jupyter Python 3.9.12
Matplotlib	Version 3.5.1
NumPy	Version 1.21.5
Pandas	Version 1.4.2
Scikit-learn	Version 1.0.2
Keras	Version 2.9.0
Tensorflow	Version 2.9.1

4.2 Evaluation Metric

The metrics used to evaluate the system for botnet attack detection include Accuracy, Precision, Recall, and F1-Score, all of which are defined by the following equations:

Accuracy = (TP+TN)/(FP+FN+TP+TN)
Precision=(TP+FP)/TP
Recall= TP/(TP+FN)
F1-Score= 2*(Precision*Recall)/(Precision+Recall)

Where:

- TP (True Positive) indicates the number of correctly identified positive instances.
- TN (True Negative) indicates the number of correctly identified negative instances.
- FP (False Positive) indicates the number of negative instances incorrectly classified as positive.
- FN (False Negative) indicates the number of positive instances incorrectly classified as negative.

4.3 Results

The experiment involved two trials in different IoT environments to evaluate the CLAE-MLP model's effectiveness in detecting botnets. Using the N-BaIoT dataset, the data was split into 70% for training and 30% for testing with Scikit-learn's dataset split function. Details of the three tested devices are provided in the next section.

4.3.1 Experiment 1 for Doorbell devices

CLAE-MLP model was applied to network data from Danmini and Ennio doorbell devices to detect various anomalies. The results of the model are shown in Tables 6(a) and 6(b). The proposed approach achieved accuracy of 92% precision, 86% recall, and 83% F1-score for detecting attack anomalies from the Danmini doorbell. Similarly, the system demonstrated its ability to identify intrusions from the Ennio doorbell, with accuracy, 83% precision, 86% recall and 85% F1-score. Figure 3 and 4 shows the confusion metrics for both danminin and ennio.

Table 6(a): Detection of different attacks from doorbell (danminin) device through CLAE-MLP model

Dorbell(Danminin)			
Attacks	Precision	Recall	F1-Score
Benign	100	100	100
Mirai_udp	96	93	95
Gafgyt combo	93	96	95
Gafgyt_junk	100	1001	100
Gafgyt scan	100	0	0
Gafgyt tcp	53	100	69
Gafgyt udp	73	99	84
Mirai ack	100	100	100
Mirai scan	100	100	100
Mirai syn	99	59	74
Mirai udpplain	100	100	100
Accuracy	87		
Weighted avg	92	87	85

Table 6(b) : Detection of different attacks from doorbell (Ennio) device through CLAE-MLP model

Dorbell(Ennio)			
Attacks	Precision	Recall	F1-Score
Benign	100	100	100
Mirai_udp	98	93	96
Gafgyt combo	94	98	96
Gafgyt_junk	100	100	100
Gafgyt scan	0	0	0
Gafgyt tcp	54	100	70
Gafgyt udp	73	99	84
Mirai ack	100	100	100
Mirai scan	100	100	100
Mirai syn	99	57	72
Mirai udpplain	100	100	100
Accuracy	88		
Weighted avg	85	88	85

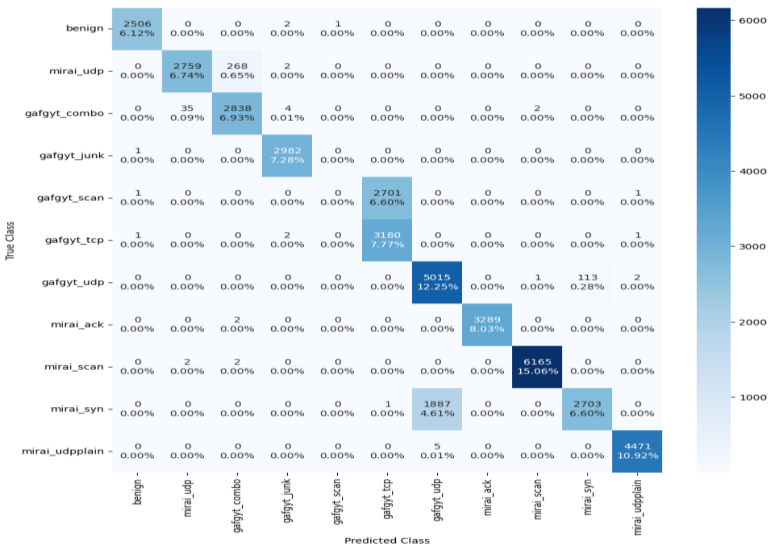

Figure 3: Confusion Metrics for Danminin

4.3.2 Experiment 2 for Baby monitor

Using network data from Baby monitor our proposed model CLAE-MLP was employed to detect various anomalies. For Baby monitor CLAE-MLP model reached an accuracy of 87%, precision of 84%, recall 87% and f1-score 85%. The results of the model is shown in Tables 7. Figure 5 shows the confusion metrics.

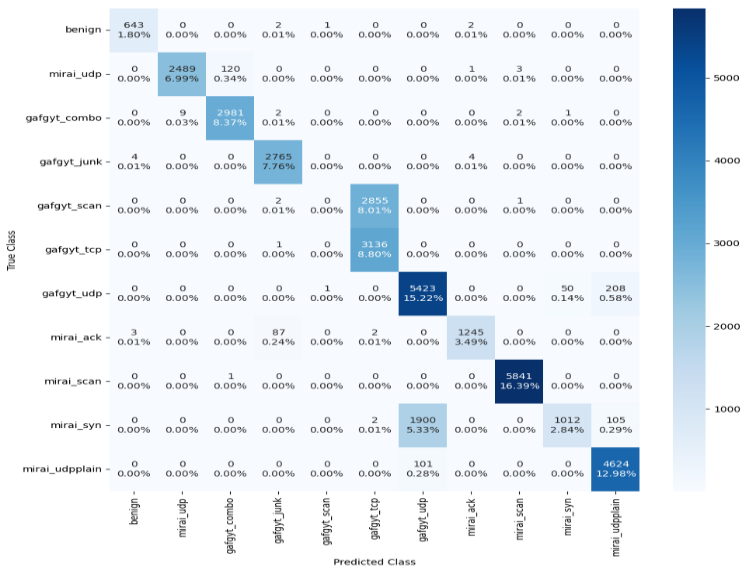

Figure 4: Confusion Metrics for Ennio

Table 7 : Detection of different attacks from Baby Monitor device through CLAE-MLP model

Baby Monitor			
Attacks	Precision	Recall	F1-Score
Benign	100	99	100
Mirai_udp	97	89	93
Gafgyt combo	90	97	93
Gafgyt_junk	99	100	100
Gafgyt scan	0	0	0
Gafgyt tcp	53	100	73
Gafgyt udp	72	99	83
Mirai ack	100	100	100
Mirai scan	100	100	100
Mirai syn	98	58	73
Mirai udpplain	100	100	100
Accuracy	87		
Weighted avg	85	87	85

5. Discussion

In this study, botnet attack is detected using the N-BaIoT dataset which employed the CLAE-MLP model. For the first experiment, the model achieved 92% on Danmini doorbell devices (Table 6(a)) and 83% on Ennio (Table 6(b)). In the second experiment, we tested on Baby Monitor device (Table 7), where we achieved 87% accuracy, but deteriorated our performance for Scan and TCP attack where 53% precision and 73% F1-score were obtained. Table 8 also compares the CLAE-MLP model's F1 score against existing systems in ensuring IoT security is still important. To this end, we evaluate results from nine IoT devices on datasets, and our results indicate that CLAE-MLP outperforms other prior CNN and LSTM-based models in most categories.

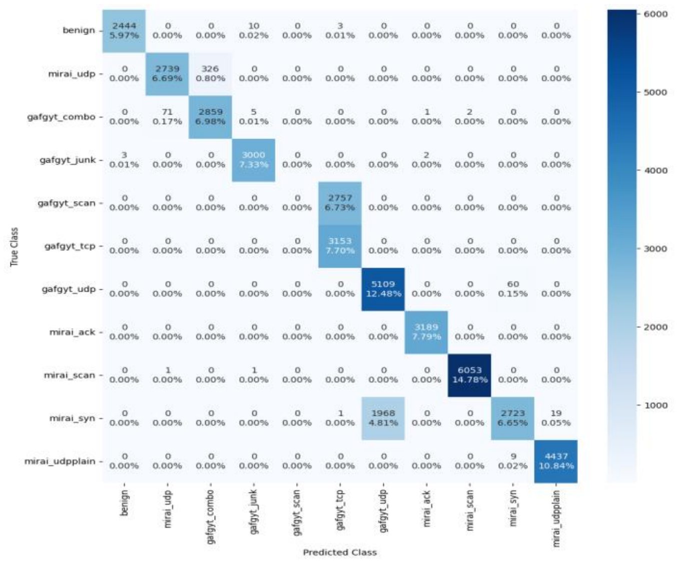

Figure 4: Confusion Metrics for Baby Monitor

Table 8. Comparision of the F1 score of the proposed model with the existing one [16]

Model	Dorbell	Baby Monitor
CNN	0.91	0.91
LSTM	0.62	0.54
CLAE-MLP(Proposed)	0.88	0.85

6. Conclusion

To detect IoT botnet attacks, we propose a CLAE-MLP encoder framework and apply it on different IoT devices. The framework also consists of a botnet dataset as well as training and detection models. We focused on the Danmini, Ennio, and thermostats judging Bashlite and Mirai attacks targeted upon, with the help of the N-BaIoT dataset which covers nine IoT devices. The junk, scan, combo, TCP flood, and UDP flood attacks comprised Bashlite; while in the case of Mirai, SYN, ACK, scan, plain UDP and UDP flood attacks were involved. Early DDoS attack detection is enhanced in this study in the context of network security. The CLAE-MLP model has a high accuracy, and future work will improve the model's performance and test it on other datasets.

References

[1] A. Shahid, M. Z. Jasni, Z. Mohamad Fadli, and I. Zakira, "A Review Paper on Botnet and Botnet Detection Techniques in Cloud Computing," 2014, Accessed: May 03, 2024. [Online].Available:https://www.researchgate.net/profile/Shahid_Anwar3/publication/283257776_A_Review_Paper_on_Botnet_and_Botnet_Detection_Techniques_in_Cloud_Computing/links/562f525308ae4742240abe a7.pdf.

[2] A. Darem, Anti-phishing awareness delivery methods, Eng., Technol. Appl. Sci. Res.11 (6) (2021) 7944–7949.

[3] Sahni, Y., Cao, J., Zhang, S., & Yang, L.: Edge mesh: A new paradigm to enable distributed intelligence in internet of things. IEEE access, 5, 16441-16458 (2017).

[4] B. K. Sovacool and D. D. F. Del Rio.: Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies. Renewable and Sustainable Energy RA. Al-Fuqaha, et al., Internet of things: a survey on enabling technologies,

[5] protocols, and applications, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2347–2376eviews, vol. 120, p. 109663, 2020.

[6] The Ultimate List of Internet of Things Statistics for 2022, https://findstack.com/internet-of-things-statistics/, last accessed: 2024/5/21.

[7] A. Holst, Number of Iot Connected Devices Worldwide 2019-2030 (2022).

[8] T. Hasan, J. Malik, I. Bibi, W. U. Khan, F. N. Al-Wesabi, K. Dev, and G. Huang, “Securing industrial internet of things against botnet attacks using hybrid deep learning approach,” IEEE Transactions on Network Science and Engineering, 2022.

[9] D. T. Son, N. T. K. Tram, and P. M. Hieu, “Deep learning techniques to detect botnet,” Journal of Science and Technology on Information security, vol. 1, no. 15, pp. 85–91, 2022.

[10] A. Alzaqebah, et al., A modified Grey Wolf optimization algorithm for an intrusion detection system, Mathematics 10 (6) (2022) 999.

[11] B. Nugraha, A. Nambiar, and T. Bauschert, “Performance evaluation of botnet detection using deep learning techniques,” in 2020 11th International Conference on Network of the Future (NoF), pp. 141–149, IEEE, 2020.

[12] M. A. Haq and M. A. Rahim Khan, “Dnnbot: Deep neural network-based botnet detection and classification.,” Computers, Materials & Continua, vol. 71, no. 1, 2022.

[13] T. H. Aldhyani and H. Alkahtani, “Attacks to automatous vehicles: A deep learning algorithm for cybersecurity,” Sensors, vol. 22, no. 1, p. 360, 2022. [21] M.Y. Alzahraniand A.M. Bamhdi, “Hybriddeep-learning modelto detect botnet attacks over internet of things environments,” Soft Computing, vol. 26, no. 16, pp. 7721–7735, 2022.

[14] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, “Hybrid deep learning for botnet attack detection in the internet-of-things networks,” IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4944–4956, 2020.

[15] M.Y.AlzahraniandA.M.Bamhdi,“Hybriddeep-learning modelto detect botnet attacks over internet of things environments,” Soft Computing, vol. 26, no. 16, pp. 7721–7735, 2022.

[16] Kim, Jiyeon, Minsun Shim, Seungah Hong, Yulim Shin, and Eunjung Choi, “Intelligent detection of iot botnets using machine learning and deep learning,” Applied Sciences, vol. 10, no. 19 pp.7009, 2020.