DEEP LEARNING MODEL FOR CYBER ATTACK DETECTION AND CLASSIFICATION IN IOT ENVIRONMENT

B. Ramesh[1], J Sri Shanthi Lasya Priya[2], T. Nandini[2], G. Yashwanth Kumar[2]

[1]Assist. Professor, [2]UG Scholar, [1,2]Department of Computer Science & Engineering (Data Science),

[1,2]Kommuri Pratap Reddy Institute of Technology, Ghatkesar, Hyderabad, Telangana.

ABSTRACT

In today's interconnected world, the widespread adoption of Internet of Things (IoT) devices has brought forth a host of conveniences and opportunities. However, this technological revolution has also opened the door to a new breed of cyber threats, with attackers exploiting vulnerabilities in IoT devices to compromise user privacy, disrupt critical services, and wreak havoc. Traditional security measures have proven inadequate to combat the evolving complexity of these cyber-attacks, necessitating a more advanced and adaptive approach. This urgency has given rise to the development of a Deep Learning Model for Cyber Attack Detection and Classification in IoT Environments (DL-IoT-CD). In addition, the need for a robust cybersecurity solution in IoT environments has become paramount due to the increasing reliance on these devices for critical applications. Existing intrusion detection systems and conventional security measures often lack the scalability and agility needed to keep pace with rapidly evolving attack techniques. As a result, there is a pressing demand for an intelligent, automated, and proactive cyber defense mechanism capable of real-time detection and classification of emerging cyber threats. The DL-IoT-CD model aims to fulfill this need by harnessing the power of deep learning algorithms to analyze vast amounts of data generated by IoT devices. By doing so, it can effectively distinguish between legitimate and malicious activities, thereby bolstering the security posture of IoT ecosystems.

Keywords: Cyber Attacks, Predictive analytics, Internet of Things, Deep learning, Artificial neural networks.

1. INTRODUCTION

The general idea of the Internet of Things (IoT) is to allow for communication between human-to-thing or thing-to-thing(s). Things denote sensors or devices, whilst human or an object is an entity that can request or deliver a service [1]. The interconnection amongst the entities is always complex. IoT is broadly acceptable and implemented in various domains, such as healthcare, smart home, and agriculture. However, IoT has a resource constraint and heterogeneous environments, such as low computational power and memory. These constraints create problems in providing and implementing a security solution in IoT devices. These constraints further escalate the existing challenges for IoT environment. Therefore, various kinds of attacks are possible due to the vulnerability of IoT devices.

IoT-based botnet attack is one of the most popular, spreads faster and create more impact than other attacks. In recent years, several works have been conducted to detect and avoid this kind of attacks [2]–[3] by using novel approaches. Hence, a plethora of relevant of relevant models, methods, and etc. have been introduced over the past few years, with quite a reasonable number of studies reported in the research domain.

Many studies are trying to protect against these botnet attacks on the IoT environment. However, there are many gaps still existing to develop an effective detection mechanism. An intrusion detection system (IDS) is one of the efficient ways to deal with attacks. However, the traditional IDSs are often not able to be deployed for the IoT environments due to the resource constraint problem of these

devices. The complex cryptographic mechanisms cannot be embedded in many IoT devices either for the same reason. There are mainly two kinds of IDSs: the anomaly and misuse approaches. The misuse-based, also called the signature-based, approach, is based on the attacks' signatures, and they can also be found in most public IDSs, specifically Suricata [4]. Formally, the attacker can easily circumvent the signature-based approaches, and these mechanisms cannot guarantee to detect the unknown attacks and the variances of known attacks. The anomaly-based systems are based on normal data and can support to identify the unknown attacks. However, the different nature of IoT devices is being faced with the difficulty of collecting common normal data. The machine learning-based detection can guarantee detection of not only the known attacks and their variances. Therefore, we proposed a machine learning-based botnet attack detection architecture. We also adopted a feature selection method to reduce the demand for processing resources for performing the detection system on resource constraint devices. The experiment results indicate that the detection accuracy of our proposed system is high enough to detect the botnet attacks. Moreover, it can support the extension for detecting the new distinct kinds of attacks.

2. LITERATURE SURVEY

Soe et al. [5] adopted a lightweight detection system with a high performance. The overall detection performance achieves around 99% for the botnet attack detection using three different ML algorithms, including artificial neural network (ANN), J48 decision tree, and Naïve Bayes. The experiment result indicated that the proposed architecture can effectively detect botnet-based attacks, and also can be extended with corresponding sub-engines for new kinds of attacks.

Ali et al. [6] outlined the existing proposed contributions, datasets utilised, network forensic methods utilised and research focus of the primary selected studies. The demographic characteristics of primary studies were also outlined. The result of this review revealed that research in this domain is gaining momentum, particularly in the last 3 years (2018-2020). Nine key contributions were also identified, with Evaluation, System, and Model being the most conducted.

Irfan et al. [7] classified the incoming data in the IoT, contain a malware or not. In this research, this work under sample the dataset because the datasets contain imbalance class. After that, this work classified the sample using Random Forest. This work used Naive Bayes, K-Nearest Neighbor and Decision Tree too as a comparison. The dataset that has been used in this research are from UCI Machine Learning Depository's Website. The dataset showed the data traffic from the IoT Device in a normal condition and attacked by Mirai or Bashlite.

Shah et al. [8] presented a concept called 'login puzzle' to prevent capture of IoT devices in a large scale. Login puzzle is a variant of client puzzle, which presented a puzzle to the remote device during the login process to prevent unrestricted log-in attempts. Login puzzle is a set of multiple mini puzzles with a variable complexity, which the remote device is required to solve before logging into any IoT device. Every unsuccessful log-in attempt increases the complexity of solving the login puzzle for the next attempt. This paper introduced a novel mechanism to change the complexity of puzzle after every unsuccessful login attempt. If each IoT device had used login puzzle, Mirai attack would have required almost two months to acquire devices, while it acquired them in 20 h.

Tzagkarakis et al. [9] presented an IoT botnet attack detection method based on a sparsity representation framework using a reconstruction error thresholding rule for identifying malicious network traffic at the IoT edge coming from compromised IoT devices. The botnet attack detection is performed based on small-sized benign IoT network traffic data, and thus we have no prior knowledge

about malicious IoT traffic data. We present our results on a real IoT-based network dataset and show the efficacy of proposed technique against a reconstruction error-based autoencoder approach.

Meidan et al. [10] proposed a novel network-based anomaly detection method for the IoT called N-BaIoT that extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic from compromised IoT devices. To evaluate the method, this work infected nine commercial IoT devices in our lab with two widely known IoT-based botnets, Mirai and BASHLITE. The evaluation results demonstrated the proposed methods ability to detect the attacks accurately and instantly as they were being launched from the compromised IoT devices that were part of a botnet.

3. PROPOSED SYSTEM

3.1 ANN

Although today the Perceptron is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of perception, seeing, and recognizing images.

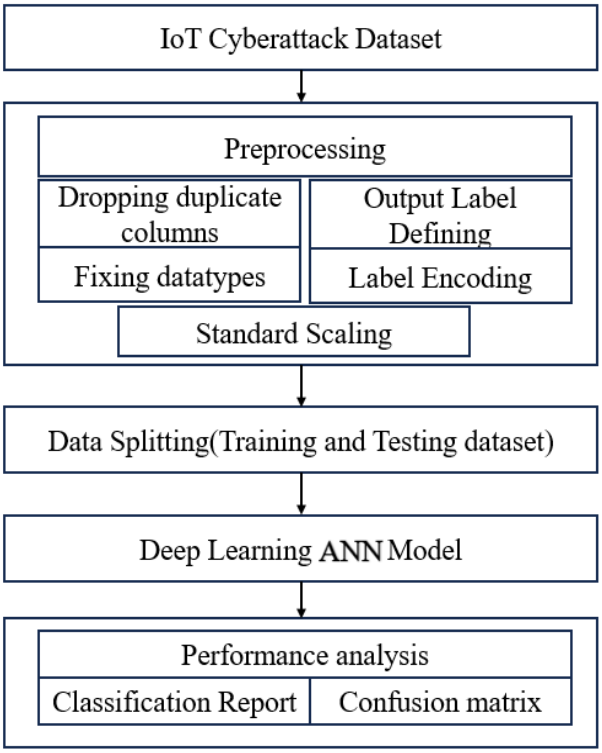
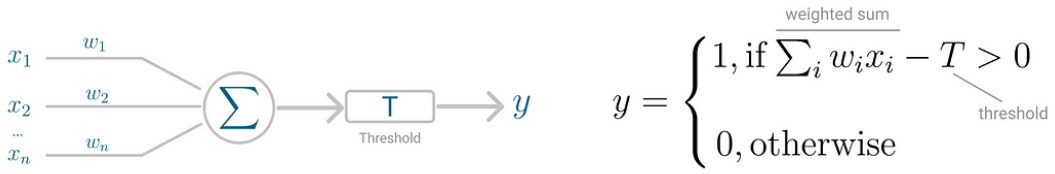


Fig. 1: Block diagram of proposed system.

In particular, interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound, temperature, etc. — the "phenomenal world" with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information. Rosenblatt's perceptron machine relied on a basic unit of computation, the neuron. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights. The major difference in Rosenblatt's model is that inputs are combined in a weighted sum and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.



Perceptron neuron model (left) and threshold logic (right).

Threshold *T* represents the activation function. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

Perceptron for Binary Classification

With this discrete output, controlled by the activation function, the perceptron can be used as a binary classification model, defining a linear decision boundary.

It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary. The perceptron loss function is defined as below:

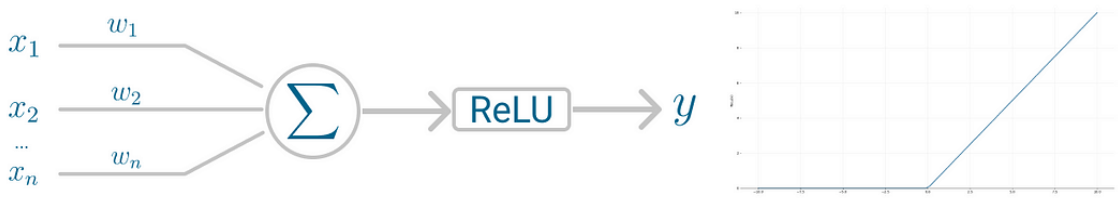
$$\underbrace{D(w,c)}_{\text{distance}} = -\sum_{i \in \text{M}} \overbrace{y_i}^{\text{output}} (x_i w_i + c)$$

misclassified observations

To minimize this distance, perceptron uses stochastic gradient descent (SGD) as the optimization function. If the data is linearly separable, it is guaranteed that SGD will converge in a finite number of steps. The last piece that Perceptron needs is the activation function, the function that determines if the neuron will fire or not. Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense! The sigmoid function maps any real input to a value that is either 0 or 1 and encodes a non-linear function. The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.

But, if you look at Deep Learning papers and algorithms from the last decade, you'll see the most of them use the Rectified Linear Unit (ReLU) as the neuron's activation function. The reason why ReLU became more adopted is that it allows better optimization using SGD, more efficient computation and is scale-invariant, meaning, its characteristics are not affected by the scale of the input.

The neuron receives inputs and picks an initial set of weights random. These are combined in weighted sum and then ReLU, the activation function, determines the value of the output.

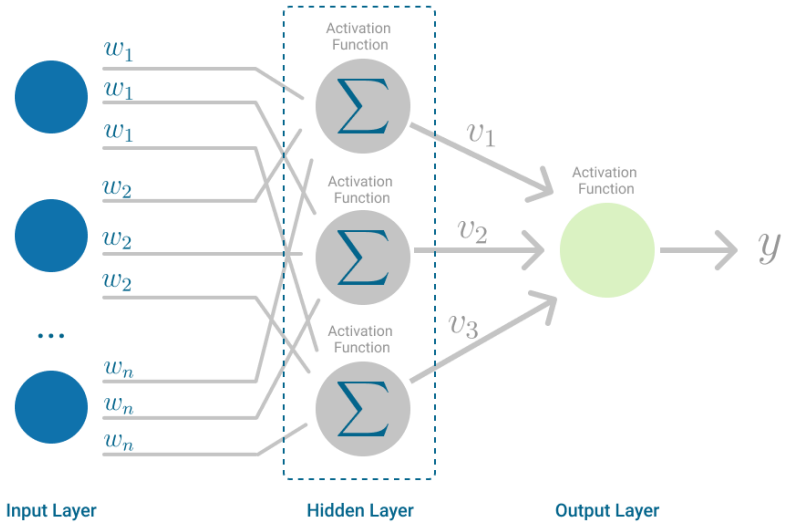


Perceptron neuron model (left) and activation function (right).

Perceptron uses SGD to find, or you might say learn, the set of weight that minimizes the distance between the misclassified points and the decision boundary. Once SGD converges, the dataset is separated into two regions by a linear hyperplane. Although it was said the Perceptron could represent

any circuit and logic, the biggest criticism was that it couldn't represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different. This was proved almost a decade later and highlights the fact that Perceptron, with only one neuron, can't be applied to non-linear data.

The ANN was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear. ANN has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a ANN can use any arbitrary activation function.



Architecture of ANN.

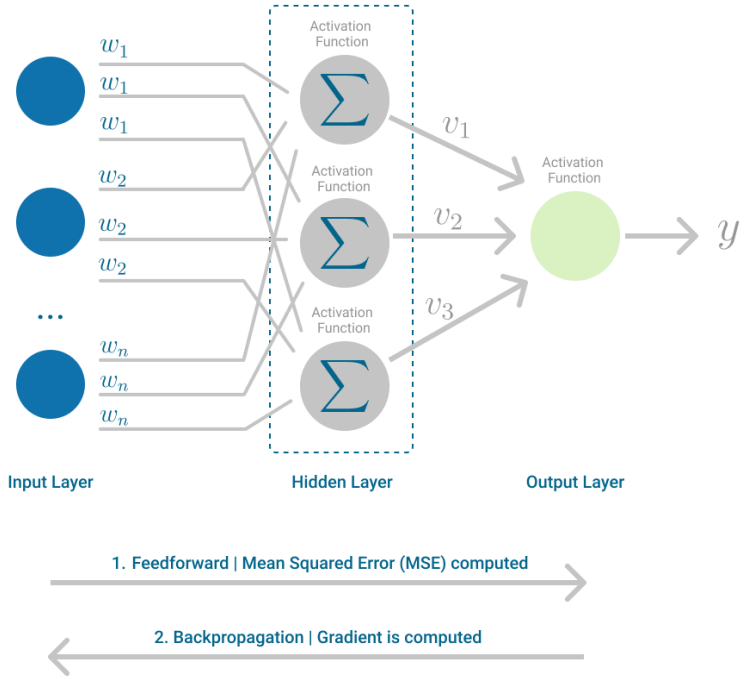
ANN falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning. This is where Backpropagation comes into play.

Backpropagation

Backpropagation is the learning mechanism that allows the ANN to iteratively adjust the weights in the network, with the goal of minimizing the cost function. There is one hard requirement for backpropagation to work properly.

The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a bounded derivative because Gradient Descent is typically the optimization function used in ANN.



ANN, highlighting the Feedforward and Backpropagation steps.

In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That's how the weights are propagated back to the starting point of the neural network. One iteration of Gradient Descent is defined as follows:

$$\underbrace{\Delta_w(t)}_{\text{Gradient Current Iteration}} = \underbrace{-\varepsilon}_{\text{Bias}} \underbrace{\frac{dE}{dw_{(t)}}}_{\substack{\text{Error} \\ \text{Weight vector}}} + \underbrace{\alpha \Delta_{w(t-1)}}_{\substack{\text{Learning Rate} \\ \text{Gradient Previous Iteration}}}$$

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn't changed more than a specified convergence threshold, compared to the previous iteration.

ADVANTAGES OF ANN

— Non-linearity: ANNs can model complex non-linear relationships between inputs and outputs, allowing them to capture intricate patterns in data that might be missed by linear models.
— Adaptability: ANNs can adapt and learn from incoming data, adjusting their parameters and internal representations to improve performance over time. This adaptability makes them suitable for tasks where the underlying data distribution may change or evolve.
— Parallel Processing: ANNs can perform computations in parallel, enabling efficient processing of large amounts of data and speeding up training and inference tasks, especially with the help of modern hardware architectures such as GPUs and TPUs.

— Fault Tolerance: ANNs can exhibit a degree of fault tolerance and robustness to noisy data. This property makes them suitable for tasks where data may be incomplete or corrupted.
— Feature Learning: ANNs can automatically learn relevant features from raw data, eliminating the need for manual feature engineering in many cases. This ability to learn hierarchical representations of data can lead to better performance on tasks such as image recognition and natural language processing.
— Generalization: ANNs can generalize well to unseen data, provided they are appropriately trained and regularized. This generalization ability allows models to perform effectively in real-world scenarios beyond the training data distribution.
— Scalability: ANNs can scale to handle large datasets and complex models with many layers and parameters. Advances in hardware and optimization algorithms have further improved the scalability of ANNs, making them applicable to big data scenarios.

4. RESULTS

Figure 2 is an illustration or visualization of a sample dataset used in your cyber-physical dataset. Figure 3 shows a tabular representation (data frame) of the features used for detecting cyber-attacks. It includes columns for various attributes or characteristics of the data. Figure 4 is a visual representation of the target variable or the column that you're trying to predict in your dataset. In the context of cyber-attack detection, this might be a binary variable indicating whether an attack has occurred or not.

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	...
0	54865	3	2	0	12	0	6	6	6.0	0.00000	...
1	55054	109	1	1	6	6	6	6	6.0	0.00000	...
2	55055	52	1	1	6	6	6	6	6.0	0.00000	...
3	46236	34	1	1	6	6	6	6	6.0	0.00000	...
4	54863	3	2	0	12	0	6	6	6.0	0.00000	...
...	...	...	...	...	...	...	...	...	...	...	...
692698	53	32215	4	2	112	152	28	28	28.0	0.00000	...
692699	53	324	2	2	84	362	42	42	42.0	0.00000	...
692700	58030	82	2	1	31	6	31	0	15.5	21.92031	...
692701	53	1048635	6	2	192	256	32	32	32.0	0.00000	...
692702	53	94939	4	2	188	226	47	47	47.0	0.00000	...

2830743 rows × 79 columns

...	min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	...	...	...	...	...	...	...	...	...	...
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	32	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN
...	20	0.0	0.0	0	0	0.0	0.0	0	0	BENIGN

Figure 2: Sample dataset used for Cyber physical dataset

	DestinationPort	FlowDuration	TotalFwdPackets	TotalBackwardPackets	TotalLengthofFwdPackets	TotalLengthofBwdPackets	FwdPacketLengthMax
0	54865	3	2	0	12	0	6
1	55054	109	1	1	6	6	6
2	55055	52	1	1	6	6	6
3	46236	34	1	1	6	6	6
4	54863	3	2	0	12	0	6
...	...	...	...	...	...	...	...
2451993	53	32215	4	2	112	152	28
2451994	53	324	2	2	84	362	42
2451995	58030	82	2	1	31	6	31
2451996	53	1048635	6	2	192	256	32
2451997	53	94939	4	2	188	226	47

2451998 rows × 77 columns

Figure 3: data frame of features used for cyber-attacks detection

```
0          BENIGN
1          BENIGN
2          BENIGN
3          BENIGN
4          BENIGN
            ...
2451993    BENIGN
2451994    BENIGN
2451995    BENIGN
2451996    BENIGN
2451997    BENIGN
Name: Label, Length: 2451998, dtype: object
```

Figure 4: Target Column of A Dataset

```
array([[ 3.54179651, -0.70409556, -0.10385474, ..., -0.17705409,
        -0.66822819, -0.62947458],
       [-0.33744755, -0.67363584, -0.06848591, ..., -0.17705409,
        -0.66822819, -0.62947458],
       [-0.33744755, -0.60000288, -0.03311707, ..., -0.17705409,
        -0.66822819, -0.62947458],
       ...,
       [ 3.76590544, -0.7040937 , -0.10385474, ..., -0.17705409,
        -0.66822819, -0.62947458],
       [-0.33935938, -0.67940758,  0.0376206 , ..., -0.17705409,
        -0.66822819, -0.62947458],
       [-0.33935938, -0.70186048, -0.03311707, ..., -0.17705409,
        -0.66822819, -0.62947458]])
```

Figure 5: features of a dataset after applying standard scalar.

Figure 5 shows the dataset's features after they've been processed using a standard scaling technique. Standard scaling is a method used to standardize the range of independent variables so they can be more easily compared. Figure 6 Is a classification report typically includes metrics such as precision, recall, F1-score, and accuracy for a classification model. This figure provides an evaluation of how well your cyber-attack detection model performs. Figure 7 is a confusion matrix is a table used in classification to summarize the performance of a classification algorithm. It shows the true positive, true negative, false positive, and false negative values. This figure provides a visual representation of these values.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	39150
1	0.99	0.74	0.85	423
2	1.00	1.00	1.00	25472
3	1.00	0.99	0.99	2075
4	1.00	0.99	1.00	34614
5	0.97	0.99	0.98	1015
6	0.99	0.99	0.99	1051
7	0.99	1.00	0.99	1136
8	1.00	1.00	1.00	2
9	0.00	0.00	0.00	10
10	1.00	1.00	1.00	11302
11	0.98	0.96	0.97	637
12	0.59	0.97	0.73	288
13	0.00	0.00	0.00	5
14	0.00	0.00	0.00	128
accuracy			0.99	117308
macro avg	0.77	0.77	0.77	117308
weighted avg	0.99	0.99	0.99	117308

Figure 6: Classification report of cyber-attack detection

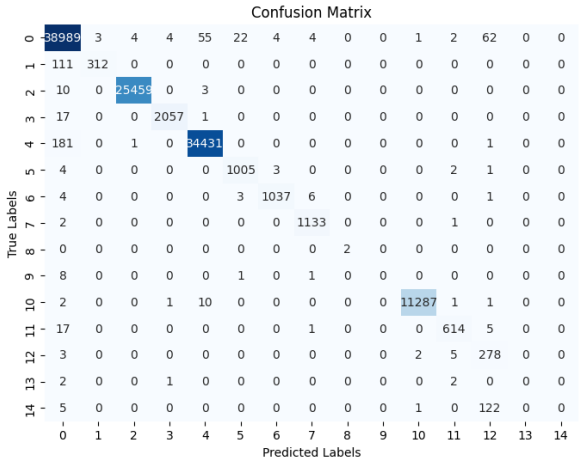


Figure 7: confusion matrix of cyber-attack detection

5. CONCLUSION

The approach of using standard scaling and a DLCNN classifier for cyber-attack detection is a promising one. It leverages advanced machine learning techniques to identify malicious activities in network traffic or system logs. By preprocessing the data effectively and training a DLCNN model, it is possible to achieve accurate and timely detection of cyber threats. However, it's important to note that the effectiveness of such a system depends on various factors, including the quality and diversity of the training data, the design of the DLCNN architecture, and the continuous monitoring and updating of the model.

REFERENCES

[1] S. Dange and M. Chatterjee, "Iot botnet: The largest threat to the iot network" in Data Communication and Networks, Cham, Switzerland:Springer, pp. 137-157, 2020.

[2] J. Ceron, K. Steding-Jessen, C. Hoepers, L. Granville and C. Margi, "Improving IoT botnet investigation using an adaptive network layer", Sensors, vol. 19, no. 3, pp. 727, Feb. 2019.

[3] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, et al., "N-baiot-network-based detection of iot botnet attacks using deep autoencoders", IEEE Pervas. Comput., vol. 17, no. 3, pp. 12-22, 2018.

[4] Shah, S.A.R.; Issac, B. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Futur. Gener. Comput. Syst. 2018, 80, 157–170.

[5] Soe YN, Feng Y, Santosa PI, Hartanto R, Sakurai K. Machine Learning-Based IoT-Botnet Attack Detection with Sequential Architecture. Sensors. 2020; 20(16):4372. https://doi.org/10.3390/s20164372

[6] I. Ali et al., "Systematic Literature Review on IoT-Based Botnet Attack," in IEEE Access, vol. 8, pp. 212220-212232, 2020, doi: 10.1109/ACCESS.2020.3039985.

[7] Irfan, I. M. Wildani and I. N. Yulita, "Classifying botnet attack on Internet of Things device using random forest", IOP Conf. Ser. Earth Environ. Sci., vol. 248, Apr. 2019.

[8] Shah, T., Venkatesan, S. (2019). A Method to Secure IoT Devices Against Botnet Attacks. In: Issarny, V., Palanisamy, B., Zhang, LJ. (eds) Internet of Things – ICIOT 2019. ICIOT 2019. Lecture Notes in Computer Science(), vol 11519. Springer, Cham. https://doi.org/10.1007/978-3-030-23357-0_3

[9] C. Tzagkarakis, N. Petroulakis and S. Ioannidis, "Botnet Attack Detection at the IoT Edge Based on Sparse Representation," 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 2019, pp. 1-6, doi: 10.1109/GIOTS.2019.8766388.

[10] Y. Meidan et al., "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," in IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, Jul.-Sep. 2018, doi: 10.1109/MPRV.2018.03367731.