Engineering and Technology Journal e-ISSN: 2456-3358

Volume 10 Issue 06 June-2025, Page No.- 5425-5431

DOI: 10.47191/etj/v10i06.14, I.F. – 8.482

Learning to Detect: An Enhanced DNN Model with Adaptive Attention for Classifying DDoS Traffic

Ahmed Saleh Khaled AL-Hurdi[1], Mohammed Fadhl Abdullah[2]

[1,2]College of engineering and computing, IT department, University of Science and Technology, Aden, Yemen.

ABSTRACT: This research proposes an enhanced Deep Neural Network (DNN) model for detecting Distributed Denial of Service (DDoS) attacks using the CIC-IDS2017 dataset. The model incorporates a Adaptive Attention Layer (AAL) and data normalization to improve feature relevance and classification accuracy. Experiments were conducted across three feature sets (78, 39, and 25), multiple dataset sizes (4K, 40K, 225K), and a range of classification thresholds (0.1 to 1.0). Results demonstrate that optimal accuracy is consistently achieved at thresholds between 0.2 and 0.4. The study confirms a positive correlation between increased feature and dataset sizes and improved detection accuracy—especially when combined with AAL and normalization. The adaptive layer significantly enhances model performance by focusing on the most informative features and reducing both false positives and false negatives. The proposed model achieved a maximum accuracy of 99.93%, outperforming several benchmark methods. These findings underscore the value of attention-based deep learning approaches in developing robust, scalable, and real-time intrusion detection systems for cybersecurity applications.

KEYWORDS: Email classification, Spam detection, Machine Learning.

1. INTRODUCTION

Cybersecurity remains a critical concern in the digital age, particularly with the increasing frequency and sophistication of network-based attacks. Among these, Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks are some of the most damaging, targeting vulnerabilities at various layers of the network protocol stack. At the network layer, attacks such as IP Spoofing, ICMP Floods, Smurf Attacks, and Routing Protocol Exploits are commonly deployed. These attacks manipulate IP headers or exploit vulnerabilities in the ICMP protocol to overwhelm targets with malicious traffic, often resulting in service disruption (Mughal, 2020). Similarly, at the transport layer, attackers exploit weaknesses in TCP and UDP protocols. For instance, TCP SYN Floods take advantage of the TCP handshake process, while UDP Floods bombard random ports to exhaust system resources. Session hijacking and attacks exploiting SSL/TLS vulnerabilities further amplify the risk at this layer (Obaid & Abeed, 2020).

Traditional signature-based Intrusion Detection Systems (IDS) are limited in their ability to detect novel or evolving threats. As a result, the cybersecurity field is increasingly embracing Artificial Intelligence and Machine Learning (ML) technologies, which offer adaptive and real-time threat detection capabilities (Ajala et al., 2024). AI, defined as the science of creating intelligent machines and software, provides the foundation for building systems that can learn and make autonomous decisions. Within AI, machine learning involves algorithms that learn patterns from data, while deep learning (DL), a subset of ML, utilizes multilayered neural networks to model complex, non-linear relationships in high-dimensional data (Sharifani & Amini, 2023).

DNNs, which consist of multiple hidden layers, have demonstrated significant success in fields such as image recognition and natural language processing. Their ability to learn hierarchical representations enables them to capture subtle patterns in complex data. Unlike conventional Artificial Neural Networks (ANNs), DNNs require substantial computational resources but offer greater predictive accuracy, especially in applications such as anomaly-based intrusion detection (Montesinos López et al., 2022).

This study proposes an enhanced DNN-based approach for detecting and classifying both DDoS and benign traffic by incorporating a Weighted Adaptive Attention Layer and Data Normalization Techniques. The model is trained and evaluated using the CIC-IDS2017 dataset, a widely-used benchmark that includes realistic traffic scenarios and a variety of attack types. The main objectives are to improve classification accuracy, reduce detection latency, and leverage adaptive learning for dynamic threat environments.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 describes the proposed model architecture, and methodology; Section 4 presents experimental results; Section 5 the discussion and the analysis of the key findings; and Section 5 concludes the work and directions for future work.

2. RELATED WORK

Numerous studies have explored machine learning (ML) and deep learning (DL) methods for intrusion detection, frequently using the CIC-IDS2017 dataset as a benchmark for evaluating model performance due to its comprehensive inclusion of modern attack vectors.

Swarnkar et al. (2021) developed an intrusion detection system optimized for big data environments by integrating dimensionality reduction technique, Principal Component Analysis (PCA) with Random Forest (RF) and K-Means clustering. Their approach reduced the dataset's original 79 features to 39, selecting 45 critical features, and achieved a detection accuracy of 99.7%. Boukhamla and Coronel (2021) enhanced the CIC-IDS2017 dataset dimensionality while preserving specificity and sensitivity. The optimized dataset enabled faster model evaluation and maintained high classification performance across number of classifiers, supporting its applicability in real-world IDS validation. Panigrahi and Borah (2018) provided a critical analysis of the CIC-IDS2017 dataset, identifying structural inconsistencies that could bias IDS performance. They proposed a refined dataset version and enhancing the reliability of detection outcomes.

Sajid et al. (2024) introduced a hybrid IDS architecture integrating ML and DL. Evaluated on CIC-IDS2017, UNSW-NB15, NSL-KDD, and WSN-DS datasets, the model achieved a test accuracy of 89.26% and F1-scores up to 98.64%, effectively capturing spatial and temporal patterns. Bandarupalli (2024) proposed a DNN-based model for network intrusion detection using 36 selected features and four hidden layers. Yulianto et al. (2019) aimed to improve AdaBoost-based IDS performance through dimensionality reduction, the system achieved an overall accuracy of 81.83%, enhancing detection efficiency.

Addressing the limitations of existing datasets, Siyyal et al. (2022) introduced a custom real-traffic-based dataset comprising over 70 features and various attack types. The dataset was evaluated using Support Vector Machine, Decision Tree, and Naive Bayes classifiers. Peng et al. (2018) proposed an improved feature selection algorithm, FACO, which combines Ant Colony Optimization with a tailored fitness function and optimized pheromone update rule. Similarly, Popoola and Adewumi (2017) combined Discrete Differential Evolution with the C4.5 algorithm to optimize feature selection, improve detection accuracy, and reduced training and testing time.

Stiawan et al. (2020) employed Information Gain to identify the most relevant features for anomaly detection. Using 52 selected features, they evaluated multiple classifiers to achieve the highest accuracy of 99.87%. Jose and Jose (2023) investigated DL models for intrusion detection in IoT environments using the CIC-IDS2017 dataset. Halimaa and Sundarakantham (2019) focused on detecting web-based attacks such as SQL Injection, Brute Force, and Cross-Site Scripting (XSS). Evaluated on the CIC-IDS2017 dataset with 78 features, their model achieved detection accuracies up to 99.57%. Finally, Singh et al. (2019) employed the Random Forest algorithm alongside Gain Ratio-based feature selection on the KDDCup99 dataset to optimize false alarm rates and improve classification accuracy.

3. METHODOLOGY

This section outlines the design, implementation, and evaluation of the proposed DNN model enhanced with a Adaptive Attention Layer and data normalization techniques for effective DDoS attack detection. The methodology comprises the model architecture, dataset preprocessing, feature selection, and performance evaluation.

3.1. Data Preprocessing and Normalization

This study uses Machine Learning CSV data, which is part of the CICIDS-2017 dataset from the ISCX Consortium. This data come within 8 traffic monitoring sessions and 78 features which labelled either as a normal traffic (defined as Benign traffic), or anomaly traffic (referred to as Attack traffic). There are 14 types of attacks in this dataset. This study focuses on the DDOS attack. Effective preprocessing is crucial for ensuring the model's accuracy and training efficiency.

The following steps are applied: the CIC-IDS2017 dataset is first loaded in CSV format, followed by data cleaning to eliminate or impute non-numeric and null values. The target column, indicating benign or DDoS traffic, is then encoded into binary labels (0 or 1). Next, min-max normalization is applied to scale all features between 0 and 1, improving training stability and convergence speed. Finally, the dataset is split into training (80%) and testing (20%) sets.

3.2. Proposed Model Architecture

The traditional Deep Neuron Networks DNN consists of three stages, namely, the Input stage or layer, the Hidden stage which usually consists of one or more layers, then the Output stage or layer. Hidden layers include the rectified linear unit function or the activation function, such as the Sigmoid function (if the output is true or false or 0, 1), the SoftMax function (if the class label has more than two pattern shown in the last stage.

Traditional DNNs apply equal weight to all input features initially, which can dilute the model's sensitivity to attack-related patterns in high-dimensional data. The proposed architecture addresses this limitation by selectively amplifying crucial signals, particularly useful in complex intrusion detection datasets like CIC-IDS2017. It enhances a traditional DNN by incorporating a Adaptive Attention Layer (AAL) between the input and hidden layers. This layer enables the model to dynamically assign importance to input features, thereby improving its ability to detect subtle patterns in network traffic. The architecture comprises four main components: an input layer that receives numerical features from the CIC-IDS2017 dataset; the AAL, which computes attention scores to reweight input features based on their

relevance; multiple hidden layers consisting of fully connected neurons with activation functions such as ReLU and Sigmoid; and an output layer that uses the Sigmoid function to perform binary classification (benign vs. DDoS attack). To explore performance trade-offs, the model is trained using different feature subsets, 78, 39, and 25 features.

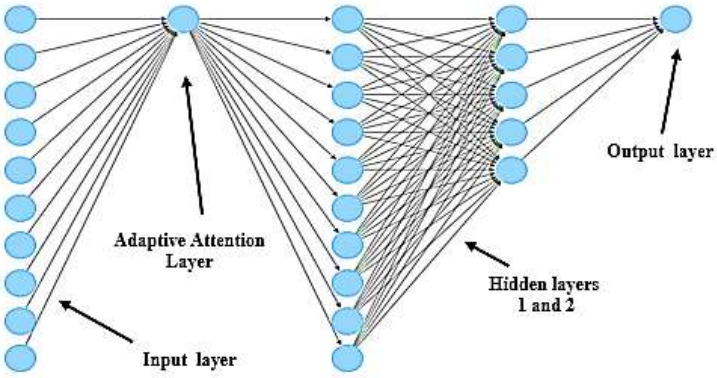

Figure 1 DNN with Adaptive Attention Layer

3.3. Adaptive Attention Layer Mechanism

The AAL operational flow includes;

(i) *Weight Computation*: During training, each input feature is assigned an initial weight wi, which is updated via backpropagation.

(ii) *Softmax Normalization*: The weights are normalized to attention scores through the softmax function.

(iii) *Feature Reweighting*: Inputs are scaled by their attention scores to prioritize informative features.

(iv) *Forward Propagation*: The reweighted feature vector proceeds through the network, influencing the final classification decision.

The **AAL** assigns importance to each feature input x_i using an attention score αi, computed via the softmax of learnable weights wi:

$$\alpha_i = \frac{e^{w_i}}{\sum_{j=1}^{n} e^{w_i}} \quad \dots (1)$$

The weighted input vector is then calculated as:

$$x_i' = \alpha_i . x_i \quad \dots (2)$$

The resulting weighted inputs x'i are forwarded to the subsequent hidden layers for further processing. This process ensures that more informative features have greater influence during forward propagation, thereby enhancing model interpretability and precision in detecting attack patterns. It also reduces overfitting by suppressing noise and irrelevant features, and significantly boosts classification metrics across various dataset sizes.

The general architecture of the proposed model consists of five phases, which are illustrated in Figure 2 above. These phases are data collection, data processing, the proposed model (DNN), training, and testing evaluation of the applied model.

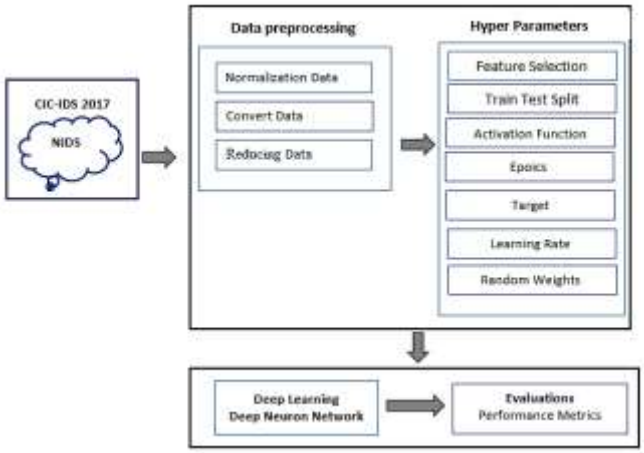

Figure 2: Proposed Model to Analysis Network Data

3.4. Training and Evaluation Process

The model training involves several stages: forward propagation, where input data passes through the attention and hidden layers to produce predictions; loss calculation, using binary cross-entropy to quantify prediction error; and backpropagation, where gradients are computed and propagated backward to update the weights via an optimizer such as Adam. The training is conducted over 50 epochs per experiment to ensure sufficient learning.

All experiments are conducted on a system equipped with an Intel Core i7 processor and 8GB RAM. The implementation and visualization utilize popular Python libraries, including Scikit-learn, NumPy, Pandas, and Matplotlib. Model performance is assessed using standard evaluation metric (Accuracy).

To examine the impact of dimensionality on model performance, the following feature sets are tested (i) Full set from the CIC-IDS2017 dataset (78 features), (ii) Reduced set based on feature relevance (39 features), and (iii) Minimal set for lightweight computation (25 features). For each scenario, the model is trained and tested with and without the AAL and normalization layer to evaluate their effectiveness.

In the beginning, data is fed into the neural network, where each input has an associated weight. The exponential values of the weights are calculated to assign relative importance to each input. These values are then normalized by dividing each exponential value by the sum of all exponential values, ensuring the weights are properly scaled. Finally, each input is multiplied by its normalized weight, determining its relative influence on the network.

Forward Propagation: In this phase, the weighted inputs are summed along with a bias term. The result is then passed through the Sigmoid activation function, which ensures that the output values remain between 0 and 1. Once the outputs are computed, they are compared with the target values, and

the error is computed as shown by Equation (3) as the difference between the actual and target values.

$$Calculated\ Error = 0.5(Target - output)^2 \quad .. \quad (3)$$

Backpropagation: After calculating the error, the network starts correcting it through backpropagation. First, the error gradient for the last node is computed based on the difference between the expected and actual output. Then, the error gradient for previous nodes is calculated based on their values. Finally, the weights are updated using the learning rate, adjusting the old weights in a way that progressively reduces the error.

This process is repeated over multiple epochs until the error is minimized, improving the accuracy of the neural network's predictions. Backpropagation is used to adjust the weights and minimize the error across the neural network.

Performance Evaluation

Accuracy: The percentage of the correctly classified objects used to calculate the classifier's accuracy is calculating accuracy is explained by Equation (4) as follows. Accuracy is the proportion of correctly classified instances:

$$Accuracy \quad = \frac{TP \ + TN}{TP + TN + FN + FP} \qquad \dots \quad (4)$$

Where TN and TP denote true negative and true positive, respectively, they are used to examine the correctness of the identified records as either positive or negative classes. At the same time, FN and FP denote false negatives and false positives, respectively. They are used to test the incorrectness of the identified records for the opposite class. The overall accuracy is 1.00, meaning that the model perfectly classified all instances in the dataset.

4. RESULTS AND ANALYSIS

Table 1 shows the DNN output with 78 features with normalization and AAL layer. Using 40K records, split data 80/20% and 50 epochs.

Table 1: Performance with 78 features, 4K records. with Normalization and Adeptive Attention layer

Threshold	TP	FP	FN	TN	Accuracy
0.1	378	57	0	365	0.92875
0.2	377	22	1	400	0.97125
0.3	375	2	3	420	0.99375
0.4	365	1	13	421	0.98250
0.5	365	1	13	421	0.98250
0.6	364	1	14	421	0.98125
0.7	364	1	14	421	0.98125
0.8	363	1	15	421	0.98125
0.9	362	1	16	421	0.98000

Table 2 shows the calculated accuracy with and without the normalization and AAL layer, and for complete use of the 78 features, split data 80% by 20% , 50 epochs, and using different data sizes.

Table 2: Accuracy With vs. Without AAL

Data Size	Threshold	Accuracy (%)	
		AAL + Norm	No AAL + No Norm
4K	0.3	99.38	98.88
40K	0.2, 0.3	99.75	83.70
225K	0.2, 0.3	99.93	64.65

Table 2 presents the model's performance analysis using 78 features, highlighting the significant benefits of normalization and the adaptive attention layer in enhancing accuracy and stability. On the small dataset of 4,000 records, the model achieved a high accuracy of 99.38% at a 0.3 threshold, demonstrating its strong class differentiation capability even with limited data.

For the medium dataset of 40,000 records, accuracy improved to 99.75% at thresholds of 0.2 and 0.3, indicating that increased data volume enabled better model adaptation and error reduction. In the large dataset of 225,000 records, accuracy further rose to 99.93% at the same thresholds, underscoring the positive impact of extensive data on learning and performance.

Conversely, removing normalization and the adaptive attention layer resulted in a gradual decline in performance. In the small dataset, accuracy decreased slightly to 98.88% across thresholds from 0.1 to 0.9, suggesting that while stability was affected, the model maintained relatively high accuracy. The medium dataset exhibited a more pronounced drop, with accuracy falling to 83.7% between thresholds 0.3 and 0.9, reflecting increased susceptibility to errors without normalization. In the large dataset, accuracy sharply declined to 64.65% at thresholds 0.2 to 0.5, highlighting the critical role of normalization in maintaining model effectiveness when handling large-scale data.

Table 3: Comparison with Different Feature Sets

Features	Threshold	TP	FP	FN	TN	Accuracy
78	0.2	3984	5	15	3965	99.75%
39	0.6	3979	5	20	3995	99.69%
25	0.4	3983	31	3	3983	99.58%

Using 78 features with normalization and an adaptive attention layer, the model achieved excellent accuracy across most thresholds. In small datasets, accuracy remained strong at low thresholds but declined at higher thresholds due to increased false negatives. For medium and large datasets, the model consistently maintained high precision, effectively leveraging the larger feature set.

However, removing normalization and the attention layer caused a noticeable performance drop. In small datasets,

accuracy became less stable with more false positives. Medium datasets saw reduced accuracy at lower thresholds, and large datasets experienced significant misclassification, highlighting the importance of normalization.

Table 3 compares performance across feature sets for a 40k data size using normalization and the adaptive attention layer. With 39 features, the model's performance was slightly lower but still strong. Normalization and the adaptive attention layer helped maintain high accuracy across thresholds, though performance declined at higher thresholds. Small datasets had manageable error rates, but medium and large datasets showed increased false negatives and greater sensitivity to threshold changes compared to the 78-feature model. Without normalization and attention, performance worsened—false positives rose in small datasets, and medium and large datasets faced more misclassification and reduced accuracy, limiting class distinction.

At 25 features, the model's accuracy decreased more noticeably. With normalization and the attention layer, performance was acceptable but weaker than models with more features. Small datasets maintained reasonable accuracy at lower thresholds but suffered at higher thresholds with more false negatives. Medium datasets showed less stability with threshold changes, and large datasets' accuracy dropped at higher thresholds. Removing normalization and attention caused sharp performance declines, with increased false positives in small datasets and significant accuracy loss and misclassification in medium and large datasets, reducing reliability.

Table 4: Best Accuracy & Threshold by Feature Set

Features	Dataset Size	Best Threshold	Accuracy (%)	FP	FN
78	225K	0.2	99.93	9	21
39	225K	0.7	99.90	3	41
25	225K	0.2	99.89	19	29

Table 4 shows the optimal threshold to the best accuracy for the different feature sets and dataset of 225k, using the normalization and the proposed layer. The performance degradation when the AAL layer avoided is shown in Table 5 for fixed 78 features and dataset size of 40k.

Table 5: Degradation Without AAL Model

Threshold	Accuracy	FP	FN	Notes
0.1–0.2	55.57%	3553	1	High FP
0.3–0.9	83.70%	0	1304	High FN at low FP

It is clearly shown that the adaptive attention layer and normalization significantly improve detection accuracy, especially for large datasets. Reducing feature count to 25 maintains high accuracy when combined with AAL plwith normalization, making the model practical for real-time or resource-constrained systems. Removing AAL or normalization leads to performance collapse, with high false positives or false negatives, especially as dataset size increases.

5. DISCUSSION

The confusion matrix is a performance measure frequently used in classification problems with two or more class-label outputs. Accuracy works using a confusion matrix. Table 6 demonstrates the best result achieved.

Table 6: Best Result Achieved

Features Size	Dataset Size	Target	Adaptive Layer	Accuracy (%)
78	4 K	0.3	True	99.37
78	40 K	0.2/0.3	True	99.75
78	225 K	0.2/0.3	True	99.93
78	4 K	0.1 to 0.9	False	98.87
78	40 K	0.3 to 0.9	False	83.69
78	225 K	0.2 to 0.5	False	64.65
39	4K	0.3	True	98.37
39	40K	0.6	True	99.68
39	225K	0.7	True	99.90
39	4K	0.1 to 0.9	False	93.12
39	40K	0.4 to 0.6	False	85.37
39	225K	0.6	False	67.12
25	4K	0.5	True	98.62
25	40K	0.4	True	99.57
25	225K	0.2	True	99.89
25	4K	0.1 to 0.9	False	97.75
25	40K	0.4	False	95.50
25	225K	0.7	False	84.96

Using 39 features with the proposed model maintain good performance, though slightly lower than with 78 features. On the small 4K dataset, accuracy reached 98.37% at a 0.3 threshold, reflecting a minor decline compared to the 78-feature model. For the 40K dataset, accuracy improved to 99.68% at a 0.6 threshold, indicating that increased data volume helped offset the reduced feature set. In the large 225K dataset, accuracy peaked at 99.90% at a 0.7 threshold, demonstrating strong performance even with fewer features when ample data is available.

Without normalization and the adaptive attention layer, accuracy notably decreased, particularly for larger datasets. Accuracy dropped to 93.12% on the 4K dataset (thresholds 0.1–0.9), 85.37% on the 40K dataset (thresholds 0.4–0.6), and further declined to 67.12% on the 225K dataset at a 0.6

threshold, indicating diminished robustness and increased errors.

Reducing features to 25 with normalization and adaptive attention resulted in a further performance decrease but remained acceptable. Accuracy was 98.62% on the 4K dataset (0.5 threshold), 99.57% on the 40K dataset (0.4 threshold), and 99.89% on the 225K dataset (0.2 threshold), suggesting that large data volume partially compensates for feature reduction. However, removing normalization and the attention layer caused more pronounced declines: 97.75% accuracy on 4K (thresholds 0.1–0.9), 95.50% on 40K (0.4 threshold), and 84.96% on 225K (0.7 threshold), underscoring the importance of these techniques for stability and accuracy.

Overall, results indicate that increasing the number of features substantially improves accuracy, with the 78-feature model outperforming others. Normalization and the adaptive attention layer are critical for enhancing stability and minimizing errors, especially in large datasets. Their absence leads to increased false positives and negatives, reducing classification reliability. Moreover, larger datasets benefit more from a higher feature count, maintaining robust accuracy across various thresholds.

Table 7: Comparison with Related Works

Researcher	Features	Algorithm	Accuracy
Bandarupalli 2024	36	DNN	0.9762
Jose & Jose 2023	71	DNN	0.9767
Swarnkar 2021	39	PCA, RF, Kmean	0.9970
Stiawan 2020	77	BN, NB, RT, J48	0.9987
Yulianto 2019	25	Adaboost	0.9001
Halimaa 2019	78	RT, DT	0.9957
Proposed Model			
Target 0.4, 0.6, 0.2	78	DNN	0.9993
Target 0.5	39	DNN	0.9990
Target 0.5	25	DNN	0.9988

Table 7 above presents a comparison between previous studies and the proposed model in terms of the number of features used, the applied algorithms, and the accuracy achieved. The results indicate that some studies, such as (BANDARUPALLI, 2024) and (Jose & Jose, 2023), used DNN networks with different feature sets but did not exceed 97.67% accuracy. Other studies, like (Swarnkar et al., 2021) and (Stiawan et al., 2020), employed multiple algorithms and achieved an accuracy of up to 99.87%.

On the other hand, our proposed model outperformed all previous works, achieving 99.93% accuracy with 78 features, 99.90% accuracy with 39 features, and 99.88% accuracy with 25 features. This demonstrates that the proposed model is highly efficient compared to previous studies, even when the number of features is reduced. Therefore, Table 39 above highlights that using DNN in the proposed model was highly effective, leading to superior results compared to traditional algorithms such as RF, PCA, and Adaboost.

6. CONCLUSION

The experimental findings of this study reveal strong interdependencies between the model's classification threshold, feature set size, dataset volume, and the inclusion of the AAL. Optimal performance was consistently achieved at threshold values between 0.2 and 0.4, where the model demonstrated the highest accuracy and lowest misclassification rates. Additionally, increasing the number of features, from 25 to 39 to 78, correlated positively with classification accuracy, highlighting the value of richer input representations. Larger datasets also enhanced performance and stability, but only when paired with proper architectural enhancements. Most notably, the inclusion of the AAL significantly improved the model's ability to focus on critical features, reduce false positives and negatives, and maintain robustness across different dataset sizes and feature configurations. This confirms that the adaptive attention mechanism, in combination with data normalization and threshold optimization, is essential for building scalable and highly accurate DDoS detection systems using deep learning.

REFERENCES

1. Ajala, O. A., Okoye, C. C., Ofodile, O. C., Arinze, C. A., & Daraojimba, O. D. (2024). Review of AI and machine learning applications to predict and Thwart cyber-attacks in real-time. *Magna Scientia Advanced Research and Reviews, 10*(1), 312-320.
2. BANDARUPALLI, G. (2024). Efficient Deep Neural Network for Intrusion Detection Using CIC-IDS-2017 Dataset.
3. Boukhamla A. K., Coronel J. (2021). CICIDS2017 Dataset: Performance Improvements and Validation as a Robust Intrusion Detection System Testbed. *International Journal of Information and Computer Security* 16(1/2):20 – 32, August 2021. DOI: 10.1504/IJICS.2021.10039325
4. Halimaa, A., & Sundarakantham, K. (2019). *Machine learning-based intrusion detection system.* Paper presented at the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI).
5. Jose, J., & Jose, D. V. (2023). Deep learning algorithms for intrusion detection systems in the Internet of Things using CIC-IDS 2017 dataset.

International Journal of Electrical and Computer Engineering (IJECE), 13(1), 1134-1141.

6. Montesinos López, O. A., Montesinos López, A., & Crossa, J. (2022). *Multivariate statistical machine learning methods for genomic prediction*: Springer Nature.
7. Mughal, A. A. (2020). Cyber Attacks on OSI Layers: Understanding the Threat Landscape. *Journal of Humanities & Applied Science Research, 3*(1), 1-18.
8. Obaid, H. S., & Abeed, E. H. (2020). DoS and DDoS attacks at OSI layers. *International Journal of Multidisciplinary Research & Pubs, 2*(8), 1-9.
9. Panigrahi R. and Borah S. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *Int. J. Eng. Technol.*, vol. 7, no. 24, pp. 479_482.
10. Peng et al. (2018) . An improved feature selection algorithm based on ant colony optimization. *IEEE Access*, vol. 6, pp. 69203_69209.
11. Popoola E. and Adewumi A (2017). Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision tree. *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 660_669.
12. Sajid, M., Malik, K. R., Almogren, A., Malik, T. S., Khan, A. H., Tanveer, J., & Rehman, A. U. (2024). Enhancing intrusion detection: a hybrid machine and deep learning approach. *Journal of Cloud Computing, 13*(1), 123.
13. Sharifani, K., & Amini, M. (2023). Machine learning and deep learning: A review of methods and applications. *World Information Technology and Engineering Journal, 10*(07), 3897-3904.
14. Singh et al. (2019). Optimization of FAR in intrusion detection system by using random forest algorithm," *SSRN Electron. J.*, vol. 5, pp. 3_6, Mar. 2019.
15. Siyyal, S. A., Khuawar, F. Y., Saba, E., Memon, A. L., & Shaikh, M. R. (2022). Analyzing ml-based IDs over real traffic. *International Journal of Innovations in Science & Technology, 4*(3), 621-640.
16. Stiawan, D., Idris, M. Y. B., Bamhdi, A. M., & Budiarto, R. (2020). CICIDS-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access, 8*, 132911-132921.
17. Swarnkar, V. K., Ambhaikar, A., & Swarnkar, S. K. (2021). Big Data Security Enhancement Based Intrusion Detection System Using K-Mean Clustering of Decomposited Features. *INFORMATION TECHNOLOGY IN INDUSTRY, 9*(1), 387-394.
18. Yulianto, A., Sukarno, P., & Suwastika, N. A. (2019). *Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset.* Paper presented at the Journal of Physics: Conference Series.

.