Hybrid Intrusion Detection using SSAE and XGBoost

Project Report

Submitted in fulfilment of the requirements

for the degree of

MASTER OF TECHNOLOGY

in

SIGNAL PROCESSING AND MACHINE LEARNING

Submitted To:

Dr. A.V. Narasimhadhan
Assistant Professor
Department of ECE
NITK Surathkal

Submitted By:

Hitesh Nayak (252SP011)
P.Sasi Priya(252SP020)
Tejas G.K.(252SP028)

Department of Electronics and Communication Engineering

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA (NITK)

SURATHKAL, MANGALORE – 575 025

Contents

1.OBJECTIVE

This project focuses on Intrusion Detection Systems (IDS) to improve the security of modern communication networks. IDS plays a crucial role in identifying unauthorized access, malicious activities, and potential cyber-attacks in real time. With the increasing complexity of security threats, traditional detection systems are no longer sufficient, motivating the need for intelligent and adaptable detection mechanisms.
Machine learning and deep learning models are implemented for accurate classification of normal and abnormal traffic. The approach enhances detection accuracy, reduces false alarms, and ensures faster response to cyber threats. This system can be applied in networking environments such as IoT, cloud infrastructure, and enterprise-level security monitoring. The work related to project are available in Github Link. Link: hyperref
`https://github.com/tejas-gkt/`
`Hybrid-Intrusion-Detection-using-SSAE-and-XGBoost`

2. INTRODUCTION

The rapid growth of network technologies and the increasing frequency of cyberattacks have made cybersecurity a critical focus in modern digital environments. Cybercriminals continuously exploit vulnerabilities to obtain unauthorized access to sensitive information, resulting in serious risks such as data theft, service disruption, and privacy breaches. With society relying heavily on interconnected computer systems and seamless internet-based services, the rate and complexity of cyber threats continue to rise. The emergence of the Internet of Things (IoT) has further expanded the attack surface, with billions of devices deployed across consumer, industrial, and governmental sectors. Although IoT enhances automation, communication, and convenience, its widespread deployment, limited computational resources, and diverse architectures make it highly vulnerable to cyber intrusions.

To strengthen cybersecurity in IoT-enabled networks, Intrusion Detection Systems (IDS) are widely employed for monitoring abnormal behaviors and detecting possible attacks. IDS can operate as Network-based (NIDS) or Host-based (HIDS), depending on the monitoring target, and utilize signature-based or anomaly-based detection strategies. While signature-based detection is effective for known threats, it fails against newly emerging attacks. Conversely, anomaly-based IDS can identify unknown threats but often suffers from high false alarm rates. Therefore, improving detection accuracy while minimizing false positives remains a key research challenge.

Machine learning (ML) has gained significant attention in IDS research due to its ability to analyze large-scale network traffic and learn patterns associated with malicious behavior. However, IoT network traffic is often high-dimensional and redundant, which negatively affects classification performance and increases storage and processing requirements.

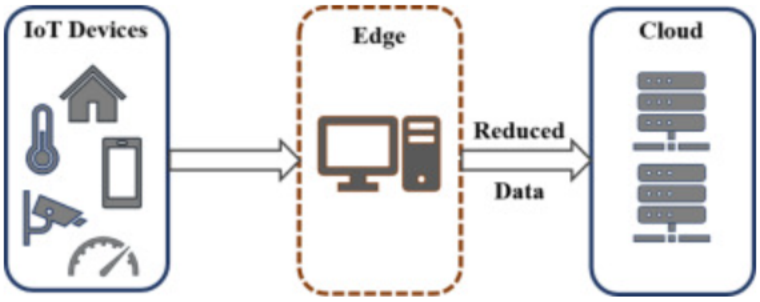

Figure 1: Edge–Cloud based intrusion detection architecture.

3.System Design and Architecture

This section presents the architecture of the proposed Intrusion Detection System (IDS), which is designed to detect cyberattacks in IoT-enabled networks efficiently. The system operates in three major phases: (1) Proposed Intrusion Detection Framework, (2) Data Preprocessing Pipeline, and (3) Feature Engineering and Dimensionality Reduction. The architecture ensures scalability, reduced communication overhead, and improved model performance using an edge–cloud computing paradigm.

3.1 Proposed Intrusion Detection Framework

The IDS framework integrates the capabilities of edge computing and cloud infrastructure to enhance security monitoring in IoT environments. IoT devices generate raw network traffic data that is first processed at the edge layer, where lightweight filtering and preliminary analysis occur. This approach reduces latency and minimizes the volume of data transmitted to the cloud.

At the cloud layer, more computationally complex operations such as training, inference, and performance evaluation are executed. The proposed model combines a Stacked Sparse Autoencoder (SSAE) for nonlinear feature learning with an XGBoost classifier for identifying attack patterns. This hybrid design enables efficient intrusion detection with improved accuracy and reduced false alarms.

3.2 Data Preprocessing Pipeline

Raw IoT traffic often contains missing values, noise, inconsistencies, and irrelevant attributes that negatively affect model learning efficiency. To address this, the data preprocessing stage includes several steps: missing value handling, noise removal, normalization, labeling, and encoding of categorical attributes.

Normalization is performed using Min–Max scaling to ensure uniform feature contribution. Removing duplicate entries reduces redundancy and prevents model overfitting. The cleaned and structured dataset resulting from this stage provides a reliable foundation for machine learning-based intrusion detection.

3.3 Feature Engineering and Dimensionality Reduction

High-dimensional traffic datasets may contain redundant or irrelevant features that increase computational complexity and degrade prediction performance. To overcome this challenge, feature engineering and dimensionality reduction techniques are applied.

The Stacked Sparse Autoencoder (SSAE) is utilized to extract compressed and meaningful feature representations. Unlike linear methods such as PCA, the SSAE captures complex nonlinear feature relationships. Reducing the feature space improves processing time, reduces memory requirements, mitigates overfitting, and enhances overall classification performance.

The reduced feature set is then fed into the XGBoost classifier, which performs intrusion detection with high accuracy, robustness, and reduced false alarm rates compared to traditional IDS approaches.

3.4Autoencoder

An Autoencoder (AE) is an unsupervised deep learning model designed to learn compressed representations of input data. It consists of two main components: an encoder, which maps the input into a lower-dimensional latent space, and a decoder, which reconstructs the original input from this compressed representation. Unlike supervised models such as CNNs or RNNs, autoencoders do not require labeled data and learn patterns through backpropagation.

The encoder reduces dimensionality by extracting essential features while discarding noise and redundant information. Hyperparameters such as number of layers, activation function, learning rate, and regularization influence model performance and are usually tuned through experimentation. Common activation functions used include sigmoid and ReLU.

The encoding process is mathematically represented as:

$$\mathbf{h} = s(W\mathbf{X} + \mathbf{b}) \tag{1}$$

while reconstruction is performed by:

$$\mathbf{Y} = s(W'\mathbf{h} + \mathbf{b}') \tag{2}$$

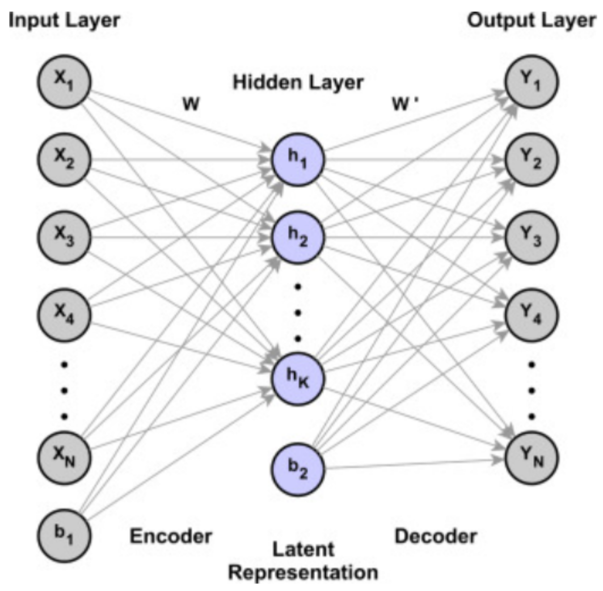

Figure 2: General architecture of an autoencoder.

where $\mathbf{X}$ is the input, $\mathbf{h}$ is the latent vector, $\mathbf{Y}$ is the reconstructed output, W and W' are weight matrices, and $s(\cdot)$ is a nonlinear activation function.

Training aims to minimize the reconstruction error using a loss function such as Mean Squared Error (MSE):

$$C = \frac{1}{m} \sum_{i=1}^{m} \mathbf{X}^{(i)} - \mathbf{Y}^{(i)\,2} \tag{3}$$

Sparse Autoencoders (SAEs) introduce a sparsity constraint to encourage only a few active neurons, improving feature interpretability and reducing overfitting. This constraint is applied using the Kullback–Leibler (KL) divergence:

$$KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{4}$$

The final loss function of an SAE combines reconstruction error, sparsity penalty, and weight regularization.

Attack Type	**Explanation**
DoS	Flooding the target with excessive traffic, causing service disruption.
Probe	Scanning the network to gather sensitive information such as open ports.
R2L	Gaining unauthorized remote access to the victim system.
U2R	Escalating privileges to gain access to confidential system resources.

Table 1: Primary forms of attacks in the dataset.

Data Type	**Training Set**	**Testing Set**
Normal	53,921	13,422
DoS	36,746	9,181
Probe	9,299	2,375
U2R	771	224
R2L	41	11
Total Instances	**100,778**	**25,195**

Table 2: Allocation of classes within the training and testing datasets.

4.Proposed Hybrid Model

Autoencoders are widely used in deep learning due to their ability to learn meaningful feature representations and reduce data dimensionality. During training, the network adjusts its weights and biases using backpropagation to minimize the difference between the original input and reconstructed output. Since the model learns without labeled data, the training process is self-supervised.

Training begins with weight initialization, followed by forward propagation through the encoder to obtain the latent representation. The decoder reconstructs the input from this encoded form. Hyperparameters such as learning rate, batch size, activation functions (e.g., sigmoid, ReLU), and number of hidden units play a critical role in the performance of the autoencoder and are tuned during training to minimize reconstruction error.

4.1 Stacked Sparse Autoencoder

Stacked Sparse Autoencoders (SSAE) extend the basic autoencoder by incorporating multiple autoencoder layers to extract increasingly abstract features. In this architecture, the encoded output of each autoencoder serves as the input to the next, enabling hierarchical feature learning.

Training is performed layer-wise rather than jointly optimizing all layers at once. Each layer is first pretrained individually, where previously learned parameters are retained while newly added layers are initialized and trained. This method improves generalization, accelerates convergence, and reduces the risk of overfitting.

A sparsity constraint is included in the loss function to ensure that only a subset of neurons remain active, preventing trivial identity mapping and improving feature selectivity. After pretraining, all layers are combined and fine-tuned using backpropagation to optimize the overall reconstruction performance.

The SSAE learns a mapping from high-dimensional input data $\mathbf{X}$ to a low-dimensional latent representation $\mathbf{Z}$, allowing the model to capture nonlinear and complex feature relationships. However, increasing depth and hidden units may lead to higher computational cost and overfitting risks. Experimental analysis showed that a four-layer SSAE achieved the best performance-to-complexity balance, as deeper models resulted in longer training times without significant performance improvements.

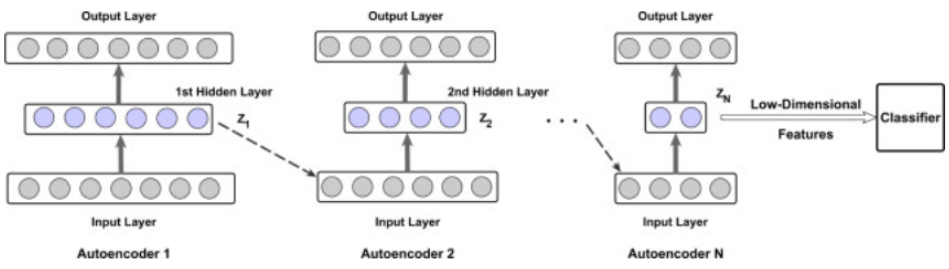

Figure 3: Architecture of the Stacked Sparse Autoencoder (SSAE) showing layer-wise feature extraction and classifier integration.

4.2 Data Preprocessing

Effective preprocessing is essential to prepare raw data for machine learning models. Since most ML algorithms cannot directly interpret categorical variables, the first step involves converting symbolic attributes into numerical form using Label Encoding and One-Hot Encoding. Label Encoding is applied when a natural ordering exists among classes, whereas One-Hot Encoding is used for unordered categorical features to avoid misleading numerical relationships.

The NSL-KDD dataset contains 41 features, including three nonnumeric fields—*protocol-type*, *service*, and *flag*. These fields are transformed using One-Hot Encoding, expanding *protocol-type* into 3 binary vectors, *service* into 70, and *flag* into 11 binary vectors, increasing the feature space from 41 to 122 dimensions.

Feature normalization is subsequently performed to ensure consistent value distribution. Due to varying value ranges across features, Min–Max normalization is applied to scale all values within the range $[0, 1]$, as shown in Eq. (5):

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{5}$$

Normalization preserves feature relationships while reducing variance, allowing the learning model to converge faster and improve prediction accuracy.

4.3 The Framework

The proposed intrusion detection model, referred to as SSA-XGB, integrates a Stacked Sparse Autoencoder (SSAE) with the XGBoost classifier. The methodology consists of two main phases: feature learning and intrusion classification.

In the first phase, the dataset undergoes preprocessing and is fed into the SSAE network. Through layer-wise training and sparsity constraints, the SSAE generates compressed low-dimensional feature representations. These features minimize redundancy, reduce training complexity, and retain essential structural information.

In the second phase, the reduced feature set is forwarded to the XGBoost classifier, which evaluates the presence of malicious activity. XGBoost is selected due to its robustness, fast computational performance, and reduced susceptibility to overfitting.

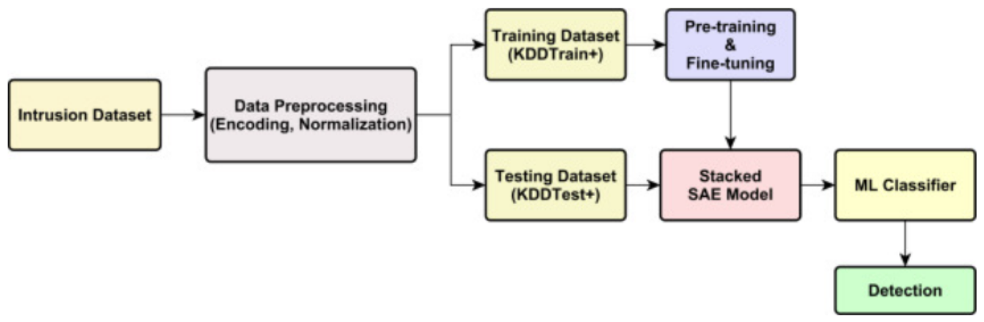
Intrusion Dataset
Data Preprocessing (Encoding, Normalization)
Training Dataset (KDDTrain+)
Pre-training & Fine-tuning
Testing Dataset (KDDTest+)
Stacked SAE Model
ML Classifier
Detection

Figure 4: Flowchart of the proposed SSA-XGB intrusion detection methodology.

5.Experimental Setup and Results

The experiments were conducted on a laptop running Windows 11 with an Intel Core i5-1035G1 CPU (3.60 GHz) and 8 GB RAM. Python version 3.8.8 was used along with relevant machine learning libraries.

Feature selection plays a critical role in improving detection performance and reducing computation time. Research indicates that models trained on reduced feature sets can achieve performance comparable to or better than models using the full feature space. In this study, the Stacked Sparse Autoencoder (SSAE) is employed to extract compact and meaningful feature representations. The depth and configuration of the SSAE significantly influence the quality of extracted features. If the network is too shallow, it may fail to capture the underlying data distribution. Conversely, excessive depth increases training cost, resource usage, and the risk of overfitting.

After preprocessing, the dataset dimensionality increased from 41 to 122 features. Accordingly, the SSAE input layer was set to 122 units. The optimal architecture was determined through experimentation and consisted of four hidden layers with neuron sizes: $[64, 32, 16, 8]$, reducing the feature space to 8 latent dimensions. Table summarizes the final parameter settings used for the SSAE model.

The dataset includes two label formats: binary (normal vs. attack) and multiclass (five attack categories). The goal of this study is to evaluate classification performance under both configurations and to examine the impact of SSAE-derived low-dimensional features on detection accuracy. The following sections present the performance evaluation results for multiclass and binary classification tasks.

Hyperparameter	Best Value
Weight decay	0.001
Learning rate	0.005
Sparsity parameter	0.05
Sparsity weight	0.1
Epochs	10
Batch size	1028

Table 3: The hyperparameter values of SSAE for binary classification.

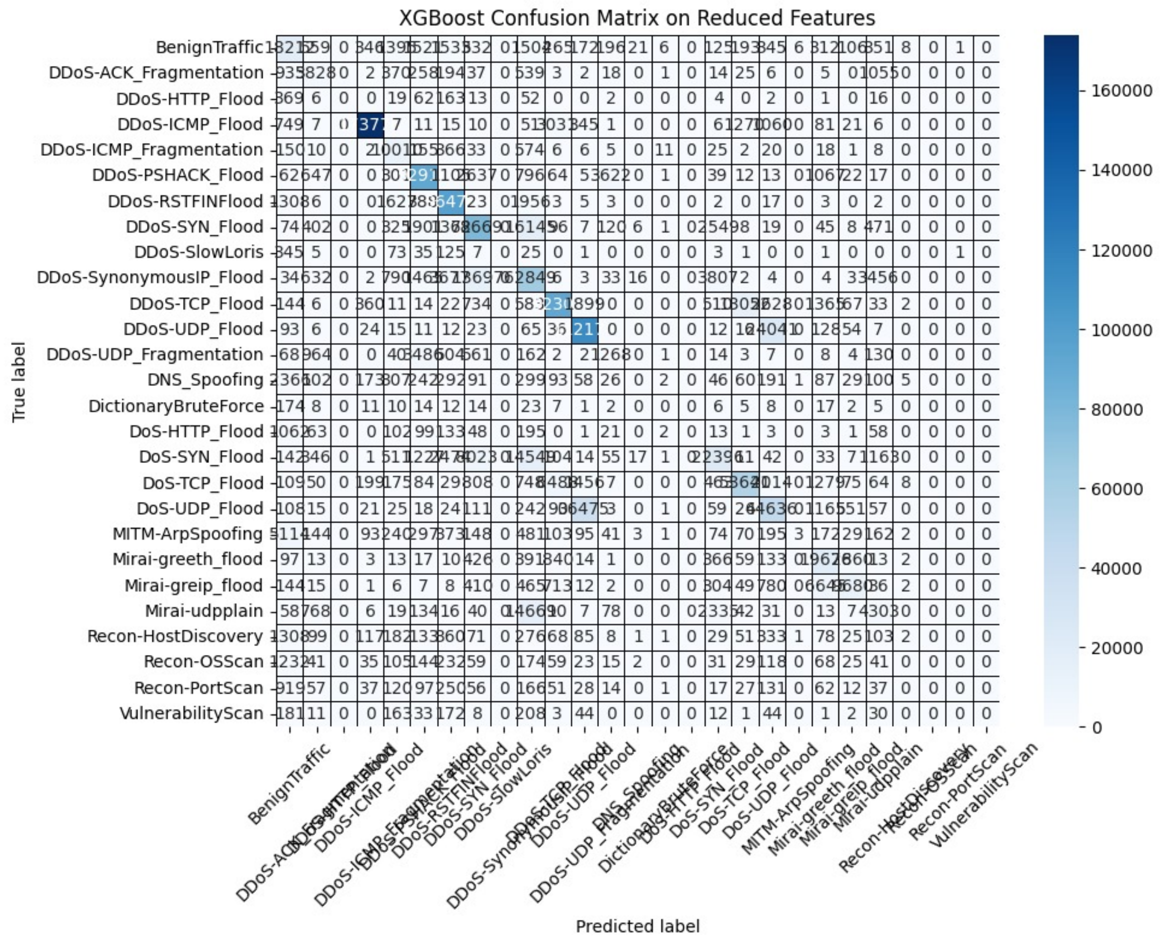

Figure 5: Reduced Image Example

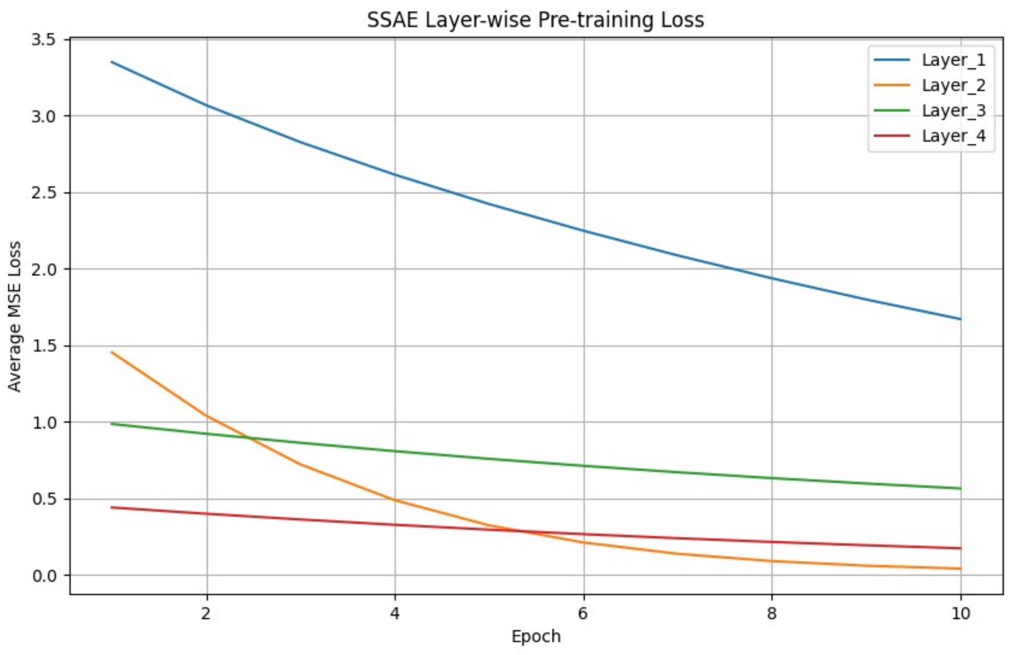

Figure 6: Reduced Image Example

6.Conclusion and Future Work

The growing complexity of modern cyber threats highlights the importance of deploying effective intrusion detection mechanisms to safeguard critical systems and data. Traditional machine learning approaches often struggle to learn meaningful feature representations from high-dimensional data, limiting their performance in large-scale cybersecurity applications.

In this study, we proposed SSA-XGB, a hybrid intrusion detection framework that combines a Stacked Sparse Autoencoder (SSAE) with an XGBoost classifier. The SSAE component performs unsupervised feature extraction and dimensionality reduction, achieving an 81% feature reduction on the NSL-KDD dataset and 83% on the CICIoT2023 dataset. These compressed latent features are then used by the XGBoost classifier, resulting in faster processing and improved model generalization. The implementation was carried out using TensorFlow and evaluated using various performance metrics, including accuracy, precision, recall, F-measure, FAR, ROC, and AUC.

While the results are encouraging, several challenges remain. Dataset imbalance and increasing data volume can affect detection reliability and model scalability. Future work will explore advanced techniques to handle imbalance, evaluate the framework on larger and more diverse datasets, and extend the approach to new threat landscapes and additional machine learning tasks.

8.References

[1] Abou El Houda, Z., Brik, B., & Khoukhi, L. (2022). “Why should I trust your IDS?”: An explainable deep learning framework for intrusion detection systems in Internet of Things networks. *IEEE Open Journal of the Communications Society*, 3, 1164–1176.

[2] Al-Garadi, M.A., Mohamed, A., Al-Ali, A.K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for Internet of Things (IoT) security. *IEEE Communications Surveys & Tutorials*, 22(3), 1646–1685.

[3] Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018). Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access*, 6, 52843–52856.

[4] Ali, S., Ghazal, R., Qadeer, N., Saidani, O., Alhayan, F., Masood, A., Saleem, R., Khan, M.A., & Gupta, D. (2024). A novel approach of botnet detection using hybrid deep learning for enhancing security in IoT networks. *Alexandria Engineering Journal*, 103, 88–97. https://doi.org/10.1016/j.aej.2024.05.113

[5] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

[6] Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232. https://doi.org/10.1214/aos/1013203451

[7] Gheni, H.Q., & Al-Yaseen, W.L. (2024). Two-step data clustering for improved intrusion detection system using CICIoT2023 dataset. *e-Prime: Advances in Electrical Engineering, Electronics and Energy*, 9, Article 100673. https://doi.org/10.1016/j.prime.2024.100673

[8] Gupta, P., Ghatole, Y., & Reddy, N. (2021). Stacked autoencoder based intrusion detection system using one-class classification. In: *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 643–648.

[9] Hindy, H., Atkinson, R., Tachtatzis, C., Colin, J.-N., Bayne, E., & Bellekens, X. (2020). Utilising deep learning techniques for effective zero-day attack detection. *Electronics*, 9(10), 1684. https://doi.org/10.3390/electronics9101684

[10] Khan, A.R., Kashif, M., Jhaveri, R.H., Raut, R., Saba, T., Bahaj, S.A., et al. (2022). Deep learning for intrusion detection and security of Internet of Things (IoT): Current analysis, challenges, and possible solutions. *Security and Communication Networks*.