Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 18s (2025)
https://www.jneonatalsurg.com

Intrusion Detection System Optimization Using ConvXGBoost for Enhanced Threat Detection

R.Usha Devi[1], Dr. R.Kannan[2]

[1]Ph.D Research Scholar, Department of Computer Science, SRMV College of Arts & Science, Coimbatore-49., usha121196@gmail.com
[2]Associate Professor, Department of Computer Science, SRMV College of Arts & Science, Coimbatore-49.

Cite this paper as: R.Usha Devi, Dr. R.Kannan, et.al (2025) Intrusion Detection System Optimization Using ConvXGBoost for Enhanced Threat Detection. *Journal of Neonatal Surgery*, 14 (18s), 143-154.

ABSTRACT
Enhancing Intrusion Detection Systems (IDS) is critical for strengthening cybersecurity against evolving threats. This research presents a comparative analysis of five machine learning algorithms such as Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), XGBoost, and Convolutional XGBoost (ConvXGBoost) for IDS classification. The evaluation is based on key performance metrics, including Accuracy, Precision, Recall, and F1-Score, across multiple attack categories such as DoS, Probe, R2L, and U2R. The experimental results indicate that ConvXGBoost outperforms other models, achieving the highest accuracy (0.97), precision (0.97), recall (0.88), and F1-score (0.93). Furthermore, the integration of Convolutional Neural Networks (CNN) with XGBoost enhances feature extraction, leading to improved classification performance. The research also presents an analysis of training performance over epochs, a confusion matrix for error assessment, and insights into model generalization. The findings highlight the potential of ConvXGBoost in optimizing IDS efficiency, offering a scalable and robust solution for cybersecurity applications.

Keywords: Intrusion Detection System, Machine Learning, Convolutional XGBoost, Cybersecurity, Classification

1. Introduction:

The rise of sophisticated cyber threats poses significant challenges to network security, necessitating robust IDS capable of identifying and mitigating attacks in real time. Traditional IDS models often fail to detect evolving attack patterns, leading to increased vulnerabilities. This research focuses on enhancing IDS performance using advanced ML techniques, evaluating five key algorithms such as RF, KNN, SVM, XGBoost, and ConvXGBoost. ConvXGBoost, integrating CNN with XGBoost, offers improved feature extraction and classification, making it a promising approach for network security [1].

The proposed approach involves preprocessing network traffic data, extracting key features, and applying ML models for classification. Performance evaluation is conducted using accuracy, precision, recall, and F1-score metrics to compare algorithm effectiveness [2]. ConvXGBoost is optimized for spatial feature extraction, enhancing its detection capabilities.

- **Enhanced IDS Performance:** The research integrates deep learning-based feature extraction with ensemble learning for improved threat detection.
- **Comparative Analysis:** A detailed evaluation of five ML algorithms provides insights into their strengths and weaknesses in handling cyber threats.
- **Optimized Feature Learning:** ConvXGBoost leverages CNNs to capture spatial correlations in network traffic, outperforming traditional models.

The findings demonstrate that ConvXGBoost achieves superior accuracy and reliability, making it a viable solution for modern cybersecurity applications.

2. LITERATURE REVIEW:

Chua et al. (2023) [3] evaluated the sustained efficacy of machine learning (ML)-based intrusion detection systems (IDS) in detecting zero-day cyber-attacks. Their approach used testing datasets generated after training datasets to account for evolving attack types and network infrastructure. They tested six ML models such as decision tree (DT), random forest (RF), support vector machine (SVM), naïve Bayes (NB), artificial neural network (ANN), and deep neural network (DNN) on the CIC-IDS2017, CSE-CIC-IDS2018, and LUFlow datasets. SVM and ANN showed the greatest resistance to overfitting, while DT and RF exhibited overfitting despite strong training performance. All models performed well when training and testing datasets were similar.

Aljuaid et al. (2024) [4] proposed a deep learning (DL)-based IDS using convolutional neural networks (CNNs) to address escalating cybersecurity threats in cloud computing. Their model incorporated dataset pre-processing, feature selection, and the SMOTE balancing method, achieving over 98.67% accuracy, precision, and recall in detecting and classifying cyber-attacks. This approach significantly enhances cloud network security. Awajan (2023) [5] developed DL-based IDS for IoT networks using a four-layer fully connected DNN architecture. The system detected attacks like Black hole, DDoS, Sinkhole, Work hole, and Opportunistic Service attacks with an average accuracy of 93.74%. It achieved an average precision of 93.71%, recall of 93.82%, F1-score of 93.47%, and detection rate of 93.21%, demonstrating its effectiveness in securing IoT networks.

Budania et al. (2023) [6] introduced DL-based IDS combining CNN, bi-directional LSTM, and encoders to monitor communication networks and detect zero-day attacks. Their model achieved a high detection rate and low false positive rate, outperforming existing models in identifying both novel and conventional attack types. Kasongo (2023) [7] proposed an ML-based IDS framework using RNN variants (LSTM, GRU, and Simple RNN) with XGBoost-based feature selection. The XGBoost-LSTM model achieved 88.13% test accuracy on the NSL-KDD dataset, while XGBoost-Simple RNN achieved 87.07% on the UNSW-NB15 dataset. The framework improved intrusion detection performance across diverse network environments.

Rushendra et al. (2021) [8] presented real-time IDS using the Residual Feedforward Neural Network (RFNN) algorithm. Tested on the NSL-KDD dataset, the system achieved 84.7% accuracy for binary classification and 90.5% for five-class classification, with detection speeds of 15 µs and 14 µs, respectively. The RFNN demonstrated potential for real-time intrusion detection. Cao et al. (2022) [9] proposed a hybrid CNN-GRU model for network intrusion detection, addressing class imbalance with ADASYN and RENN. The model achieved classification accuracies of 86.25%, 99.69%, and 99.65% on the UNSW-NB15, NSL-KDD, and CIC-IDS2017 datasets, respectively, outperforming comparable models.

Thirimanne et al. (2022) [10] developed a Real-Time Intrusion Detection System (RT-IDS) using DNN trained on 28 features from the NSL-KDD dataset. The system achieved 81% accuracy, 96% precision, 70% recall, and an 81% F1-score, demonstrating its effectiveness in real-time intrusion detection. Azar et al. (2023) [11] proposed four hybrid IDS for satellite-terrestrial integrated networks (STINs), combining RF, LSTM, ANN, and GRU with sequential forward feature selection (SFS). The RF-SFS-GRU model achieved 87% accuracy on the STIN dataset and 79% on UNSW-NB15, highlighting the importance of feature selection in improving detection accuracy. Asgharzadeh et al. (2024) [12] introduced enhanced IDS for IoT using CNN and a binary multi-objective enhanced Gorilla Troops Optimizer (BMEGTO) for feature selection. The CNN-BMEGTO-KNN model achieved 99.99% and 99.86% accuracy on the NSL-KDD and TON-IoT datasets, respectively, demonstrating its effectiveness in IoT anomaly detection.

This review highlights advancements in IDS using ML and DL techniques, focusing on zero-day attack detection, cloud and IoT security, and real-time intrusion detection. Key contributions include hybrid models, feature selection methods, and innovative architectures like CNN-GRU and DNN, which have significantly improved detection accuracy and efficiency across diverse network environments.

3. MATERIALS AND METHODOLOGY

The proposed IDS framework integrates advanced machine learning for cyber threat detection. It begins with data collection from cybersecurity datasets, ensuring diverse attack scenarios. Preprocessing handles missing values, normalizes data, encodes categorical features, and filters noise. Feature selection extracts key attributes to enhance classification. The model, trained on machine learning algorithms, classifies network traffic as normal or malicious, identifying attack types like DoS, Probe, R2L, and U2R [13]. An alert mechanism notifies security teams, while automated mitigation applies security updates and blocking. The framework ensures high accuracy, fewer false positives, and adaptability to evolving threats.

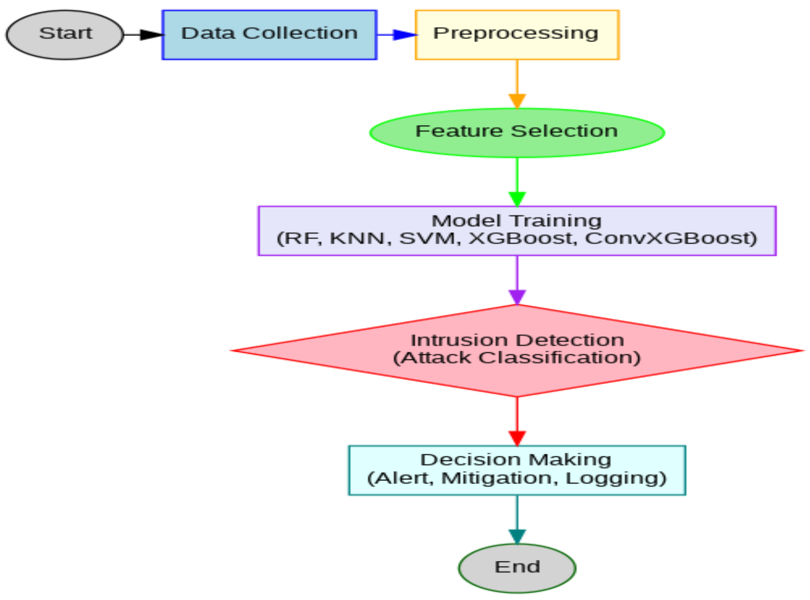


Fig.1. Machine Learning-Based Intrusion Detection System Architecture

The above figure presents a flowchart of an intrusion detection system, detailing the process from data collection to decision-making using machine learning models.

3.1. Data Description:

The NSL-KDD dataset is an improved iteration of the KDD CUP 99 dataset, extensively utilized for research in intrusion detection. It removes redundant entries, guaranteeing impartial assessment and enhanced classifier efficacy [14]. The dataset comprises 41 features, encompassing nominal variables such as Protocol type, Service, and Flag, as well as numeric (double) attributes including duration, src_bytes, dst_bytes, and numerous connection statistics. Records are categorized as either normal (no attack) or as specific attack kinds, including nmap, Neptune, and multichip, with supplementary levels denoting the severity of the attacks. Its equitable form renders it a dependable standard for assessing intrusion detection systems.

Table.1. Feature Description

Feature Type	Features
Nominal Features	Protocol type, Service, Flag
Binary Features	Land, Logged_in, Root_shell, Su_attempted, Is_host_login, Is_guest_login
Numeric Features	Duration, Src bytes, Dst bytes, Wrong fragment, Urgent, Hot, Num failed logins, Num_compromised, Num root, Num file creations, Num shells, Num access files, Num outbound_cmds, Count, Srv_count, Serror_rate, Srv_serror_rate, Rerror_rate, Srv_rerror_rate, Same_srv_rate, Diff_srv_rate, Srv_diff_host_rate, Dst_host_count, Dst_host_srv_count, Dst_host_same_srv_rate, Dst_host_diff_srv_rate, Dst_host_same_src_port_rate, Dst_host_srv_diff_host_rate, Dst_host_serror_rate, Dst_host_srv_serror_rate, Dst_host_rerror_rate, Dst_host_srv_rerror_rate

The `attack categories` dictionary maps various attack names to their respective categories, such as 'DoS', 'Probe', 'R2L', and 'U2R'. Each key represents a specific attack type (e.g., 'normal', 'back', 'land'), while the corresponding value indicates its classification.

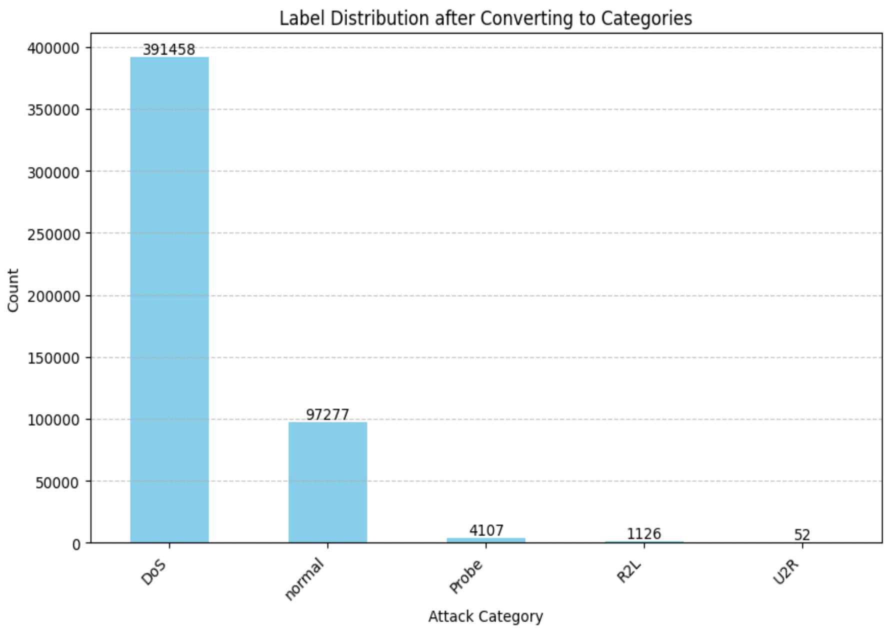


Fig.2. Label Distribution after Converting to Categories

The above bar chart illustrates the distribution of labels across different categories. The 'normal' category is the most prevalent, with nearly 391,500 labels, while the 'U2R' category is the least frequent, containing only 52 labels. Overall, the dataset comprises over 400,000 labels distributed across five categories.

3.2. Pre-processing

3.2.1. One-Hot Encoding

It is used to convert categorical features, such as protocol types, service names, and flags, into numerical vectors suitable for machine learning models. For example, a categorical feature like **Protocol** with values {TCP, UDP, ICMP} can be encoded into binary vectors where each unique category is represented as a vector with a 1 at the index corresponding to the category and 0 elsewhere. Specifically, if X_i is the categorical feature and C_j is a specific category, the one-hot encoded vector is defined as:

$$\boldsymbol{One - Hot\ Encoding(X_i = C_j) = \begin{cases} 1 & if\ \ X_i = C_j \\ 0 & otherwise \end{cases}}$$

This transformation helps machine learning algorithms in IDS by representing categorical attributes in a numerical format, enabling the detection of intrusions based on patterns in the encoded data [15].

3.2.2. Data normalization

Data normalization is a crucial pre-processing step in an Intrusion Detection System (IDS) to ensure that numerical features are on a similar scale, which is especially important for machine learning algorithms. In IDS, features such as packet size, duration, and byte counts can have vastly different ranges, and if one feature has a much larger scale, it could dominate the model's learning process. A common method of normalization is **min-max normalization**, which scales the values of each feature to a range between 0 and 1 using the formula:

$$\boldsymbol{X' = \frac{X - X_{min}}{X_{max} - X_{min}}}$$

Where, $\boldsymbol{X}$ is the original value of the feature, $\boldsymbol{X_{min}}$ is the minimum value, $\boldsymbol{X_{max}}$ is the maximum value, and $\boldsymbol{X'}$ is the normalized value. The process involves identifying the features that need normalization, calculating their minimum and maximum values, and applying the normalization formula to scale each data point to the [0, 1] range. This ensures that all features contribute equally to the model, preventing any single feature from disproportionately influencing the model's performance and improving the overall efficiency of the IDS [16].

3.3. Feature Analysis

This section visualizes the distribution of selected features across attack categories ('DoS', 'Probe', 'R2L', 'U2R') using box plots. Each subplot represents a feature, highlighting variations in its values across different attack types. This comparative analysis helps identify patterns and anomalies, offering insights into feature behaviour in network attacks. The grid layout ensures clear interpretation of feature-attack relationships.

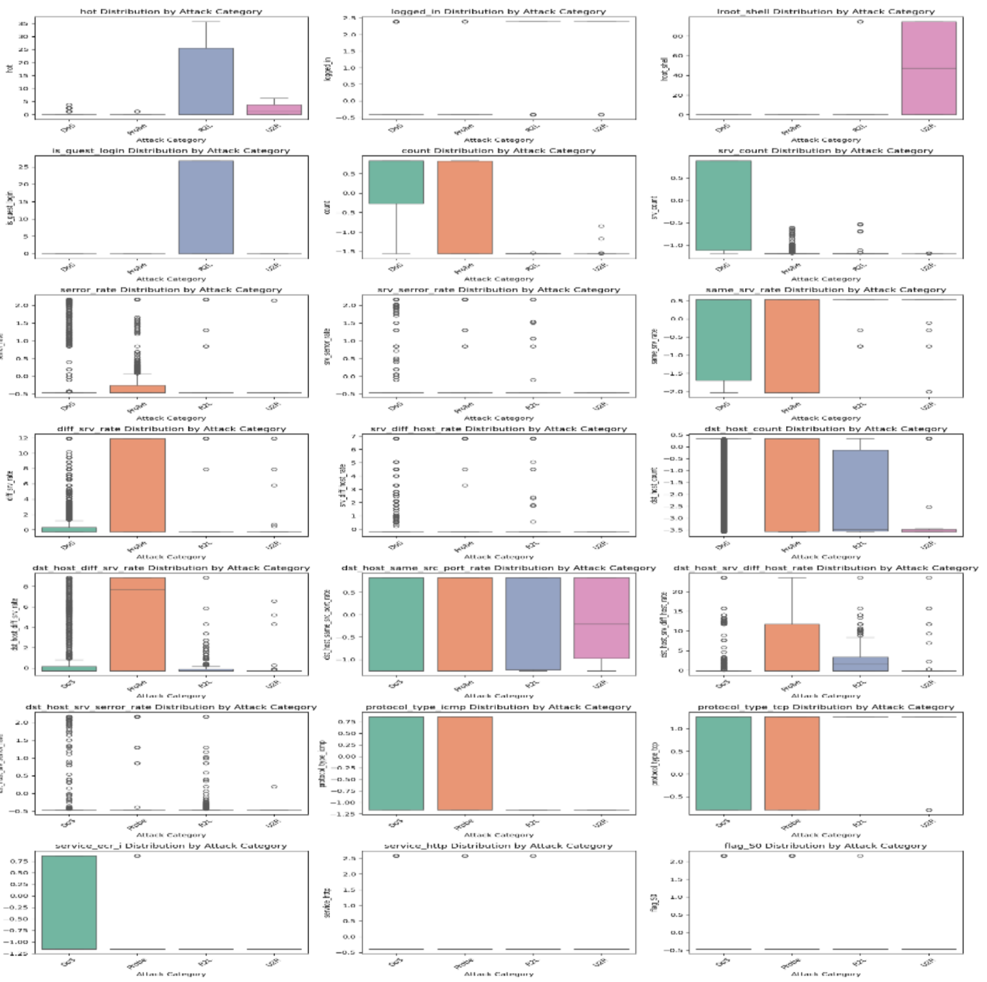


Fig.3. Feature Distribution across Attack Categories: Box Plot Analysis

The above image contains multiple box plots visualizing feature distributions across attack categories in an IDS dataset (KDD Cup 99/NSL-KDD). Key features like root_shell, hot, srv_count, and serror_rate show distinct patterns, aiding in attack detection and feature selection.

3.4. Classification

Classification in IDS uses various ML algorithms to detect threats. RF builds multiple decision trees using bootstrap sampling and aggregates predictions via majority voting. KNN classifies data based on the majority class of its k-nearest neighbors using distance metrics. XGBoost sequentially constructs decision trees, minimizing loss with gradient boosting and regularization. SVM find an optimal hyperplane to separate classes, using kernels for non-linear data. ConvXGBoost combines CNNs for feature extraction and XGBoost for classification, leveraging both spatial pattern recognition and robust prediction. These algorithms enhance IDS by addressing overfitting, feature selection, and non-linear separability.

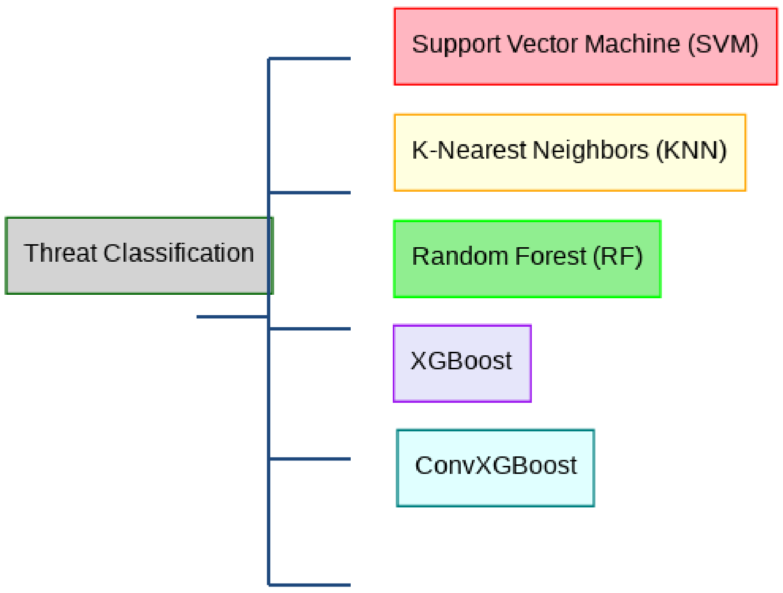


Fig.4. IDS Classification with ML Algorithms

The above figure illustrates machine learning models employed for threat classification, encompassing SVM, KNN, RF, XGBoost, and ConvXGBoost [17].

3.4.1. Random Forest:

Random Forest is a supervised ML algorithm commonly used for IDS. It constructs multiple decision trees from various samples of training data and aggregates their predictions through majority voting for classification tasks, enhancing detection accuracy and robustness against over fitting compared to single decision trees.

i. **Input:** Dataset $\boldsymbol{D}$ with mmm features and $\boldsymbol{N}$ samples, Number of trees $\boldsymbol{T}$ and features per split $\boldsymbol{M}$ (where $M < m$)

ii. **Construct Random Forest**: For **t=1 to T**

a. **Sample Selection**: Create bootstrap sample $\boldsymbol{D_t}$ by randomly selecting $\boldsymbol{N}$ samples from $\boldsymbol{D}$ with replacement.

b. **Feature Selection**: Randomly select **M** features from **m**

c. **Decision Tree Construction**: Build tree $\boldsymbol{T_t}$ using $\boldsymbol{D_t}$ and selected features. At each node, use Gini impurity to find the best split:

$$\boldsymbol{Gini(t) = 1 - \sum_{i=1}^{c} p(i|t)^2}$$

iii. **Prediction**: For a new sample $\boldsymbol{x}$**,** each tree predicts $\boldsymbol{\hat{y}_t}$. Final prediction $\boldsymbol{\hat{y}}$ majority voting:

$$\boldsymbol{\hat{y} = arg \max_{y \in classes} \sum_{t=1}^{T} I(\hat{y}_t = y)}$$

3.4.2. K-Nearest Neighbours (KNN)

KNN in IDS is a classification method that detects potential security threats by categorizing network connections according to the behaviours of their nearest neighbours in a multi-dimensional feature space. It use distance measures to evaluate similarity and classifies an unknown data point through majority vote among the k nearest distances [18].

i. **Data Representation**: Each network connection or packet is depicted as a point within a multi-dimensional feature space, with each feature reflecting attributes such as protocol type, IP addresses, and packet size

ii. **Distance Metric**: The Euclidean distance is commonly used to measure proximity:

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{n} (x_{ip} - x_{jp})^2}$$

iii. **Choosing k**: The parameter ***k*** specifies the number of nearest neighbours taken into account. A diminutive ***k*** is susceptible to noise, whilst an extensive k may excessively smooth outcomes.

iv. **Classification Process**:

- **Compute Distances:** Calculate the distances between the new data point x and all training points.
- **Choose Neighbours:** Determine the ***k*** closest neighbours.
- **Electoral Participation:** Assign the class to ***x*** according to the predominant class among the neighbours:

$$C(x) = mode\left(C(x_1), C(x_2), \dots C(x_k)\right)$$

3.4.3. XGBoost (Extreme Gradient Boosting)

XGBoost is an ensemble learning method that integrates several weak learners, usually decision trees, to formulate a robust prediction model. It employs a gradient boosting framework, constructing trees consecutively, with each tree endeavouring to rectify the flaws of its predecessors. XGBoost is a robust machine learning method frequently employed for classification problems, including IDS.

i. **Objective Function**: The goal function in XGBoost is defined as:

$$L = \sum_{i=1}^{n} L\,(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Let ***L*** denote the overall loss function, n represent the number of training examples, y_i signify the true label for the i-th instance, $\hat{y}_i$ indicate the predicted label for the i-th instance, ***L*** be a loss function (e.g., logistic loss for binary classification), ***K*** denote the number of trees, f_k represent the k-th tree, and $\Omega(f_k)$ serve as a regularization term to regulate the model's complexity.

ii. **Tree Construction**: In XGBoost, trees are incrementally added by minimizing the loss function according to the following formula:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Where, $\hat{y}_i^{(t)}$ is the prediction after ***t*** trees, $f_t(x_i)$ is the output of the t-th tree for input x_i .

iii. **Gradient and Hessian**: The updates are derived from the gradient and the second derivative (Hessian) of the loss function:

$$g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$$

$$h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$$

iv. **Final Prediction**: The conclusive prediction is derived by aggregating the predictions from all the trees:

$$\hat{y} = \sum_{t=1}^{T} f_t(x)$$

Where, ***T*** is the total number of trees [18].

3.4.4. Support Vector Machines (SVM)

SVM are robust supervised learning algorithms employed for classification tasks, including IDS. SVMs demonstrate notable efficacy in high-dimensional domains, rendering them appropriate for network traffic monitoring. SVMs operate by identifying a hyper plane that optimally distinguishes data points belonging to disparate groups. The objective is to optimize the separation between the nearest points (support vectors) of each class [18].

i. **Hyper plane**: In an n-dimensional space, a hyper plane can be expressed as:

$$w.x + b = 0$$

In this context, ***w*** represents the weight vector (perpendicular to the hyper plane), x is the input feature vector, and ***b*** signifies the bias term.

ii. **Maximizing the Margin**: The objective of SVM is to optimize the margin ***M,*** which is defined as the distance from the hyper plane to the closest data point of either class. The margin is defined as:

$$M = \frac{2}{||w||}$$

Consequently, maximizing the margin is synonymous with minimizing $||w||^2$

iii. **Objective Function**: The optimization problem can be articulated as: $\min_{w,b} \frac{1}{2}||w||^2$

Contingent upon the limitations: $y_i(w.x_i + b) \geq 1 \quad \forall i$

Here, y_i represents the true label of the i-th instance, assuming values of +1 or -1.

iv. **Kernel Trick**: To address non-linearly separable data, SVM employs the kernel trick to transform the input space into a higher-dimensional space. The decision function is expressed as:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b$$

Where, α_i are the Lagrange multipliers, is the kernel function (e.g., linear, polynomial, radial basis function).

v. **Decision Rule**: The ultimate determination for a new data point x is expressed as:

$$Predict(x) = \begin{cases} +1 \; if \; f(x) > 0 \\ -1 \; if \; f(x) \leq 0 \end{cases}$$

3.4.5. ConvXGBoost (proposed)

The convXGBoost technique integrates CNNs with XGBoost for IDS. The objective is to employ CNNs to extract spatial characteristics from network traffic data, thereafter transmitting these features to the XGBoost classifier for final prediction.

i. **CNN Feature Extraction:** CNN Extraction of Features: The input of the CNN is generally network traffic data, frequently represented as multi-dimensional feature matrices. The convolutional layers utilize filters to identify spatial patterns within the data. The convolution process in each layer is expressed as:

$$Z^{(l)} = f\left(W^{(l)} * X^{(l-1)} + b^{(l)}\right)$$

$Z^{(l)}$ represents the output of the l-th layer, $W^{(l)}$ denotes the filter/kernel utilized at layer l, * signifies the convolution operation, $X^{(l-1)}$ is the input to the current layer from the preceding layer, $b^{(l)}$ is the bias term, and, $f(.)$ is the activation function (e.g., ReLU).

ii. **Pooling and Flattening:** Following the convolutional layers, a pooling layer decreases the dimensionality, and the feature map is converted into a vector. The vector is subsequently transmitted to the XGBoost model.

iii. **XGBoost Classifier:** XGBoost operates by constructing an ensemble of decision trees through a gradient-boosting architecture. The optimization method seeks to minimize a loss function. The objective function for XGBoost is:

$$L(\theta) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

Where, $l(\hat{y}_i, y_i)$ represents the loss function (e.g., log loss for classification), and $\Omega(f_k)$ denotes the regularization term for each tree. f_k, $\hat{y}_i$ represents the projected output, y_i while denotes the true label.

XGBoost employs a second-order Taylor expansion to estimate the loss and enhance tree construction.

$$L^{(t)} \approx \sum_i \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) + \Omega(f_t) \right]$$

In this context,g_i and h_i represent the first and second derivatives of the loss function, respectively, whereas f_t denotes the prediction from the current tree.

iv. **Conclusive Result:** The ultimate result of convXGBoost is the classification label for the network traffic sample, denoting whether it is normal or indicative of an intrusion.

This hybrid model utilizes CNN's capacity to identify intricate data patterns and XGBoost's proficiency in robust classification, rendering it effective for intrusion detection applications.

Pseudo code for ConvXGboost:

```
Step 1: Prepare the data.
Input: Network traffic data (X), Labels (Y)
Standardize X and partition into training and testing datasets: $(x_{train}, x_{test}, y_{train}, y_{test})$
Step 2: Convolutional Neural Network for feature extraction
Function CNN Feature Extractor(X):
    Implement convolutional layers followed by ReLU activation and pooling layers.
    Render the output in a flat format
    Return flattened characteristics (F)
Step 3: Train XGBoost classifier
For every batch in $X_{train}$:
    Derive characteristics $F_{train}batch$= CNN Feature Extractor ($X_{train}batch$)
```

```
Train XGBoost (XGB) on $F_{train}batch$, $y_{train}$
Step 4: Assessment and Evaluation
For each batch in $x_{test}$:
    Derive characteristics $F_{test}batch$ = CNN Feature Extractor($X_{test}batch$)
    Predict $Y_{pred}$= $XGB.Predict = F_{test}batch$
Assess the model utilizing $Y_{pred}$ and $y_{test}$
Return: Assessment metrics and trained model
```

4. RESULT AND DISCUSSION

The results and discussion section presents a comparative evaluation of various machine learning algorithms applied to intrusion detection systems. It analyses the performance of RF, KNN, SVM, XGBoost, and ConvXGBoost using key metrics such as accuracy, precision, recall, and F1-score. Through detailed tables and visualizations, this section highlights the strengths and limitations of each algorithm, demonstrating how ConvXGBoost achieves superior detection capabilities. Additionally, training performance, confusion matrix insights, and model effectiveness are discussed to emphasize the impact of advanced ML techniques on cybersecurity.

4.1. Performance Metrics

Performance metrics such as Accuracy, Precision, Recall, and F1-Score [18] assess the effectiveness of IDS by measuring its ability to correctly classify cyber-attacks while balancing detection and false alarms.

Accuracy: Accuracy in Cyber Attacks pertains to the capability of IDS to accurately categorize network data as either malicious (attack) or benign (normal). Accuracy is determined by the ratio of correctly detected instances (true positives and true negatives) to the total occurrences, expressed by the formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where, TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. Although accuracy serves as a broad indicator of system efficiency, it may be deceptive in imbalanced datasets with limited attack samples, rendering precision, recall, and F1-score more pertinent in certain contexts.

Precision: Precision in Cyber Attacks quantifies the ratio of accurately recognized attacks (True Positives) to the total instances categorized as attacks (True Positives + False Positives). It signifies the reliability of the system in categorizing traffic as an attack. The equation for precision is:

$$Precision = \frac{TP}{TP + FP}$$

Where, TP (True Positives) refers to accurately identified attacks, while FP (False Positives) denotes legitimate traffic erroneously classed as assaults. High precision signifies a reduction in false alarms, which is essential for minimizing unnecessary alerts in IDS.

Recall: Recall in Cyber Attacks assesses the capability of an Intrusion Detection System (IDS) to accurately identify genuine attacks. The ratio of accurately diagnosed attacks (True Positives) to the total number of real attacks (True Positives + False Negatives). The equation for recall is:

$$Recall = \frac{TP}{TP + FN}$$

In this context, TP (True Positives) refers to accurately identified attacks, while FN (False Negatives) denotes attacks that were not detected. High recall guarantees the detection of the majority of attacks, rendering it essential for reducing overlooked dangers in cyber security.

F1-Score: The F1-Score in cyber-attacks is the harmonic mean of precision and recall, offering a balanced assessment of an Intrusion Detection System's performance, particularly in instances of class imbalance. The F1-Score is computed using the formula:

$$F1 - Score = \frac{2.(Precision.Recall)}{Precision + Recall}$$

Precision assesses the accuracy of attack predictions, whereas recall evaluates the system's capacity to identify all genuine attacks. An elevated F1-score signifies an effective equilibrium between detecting the majority of attacks (recall) and reducing false positives (precision).

Experimental Analysis

The experimental analysis evaluates the performance of various machine learning algorithms for intrusion detection, comparing their accuracy, precision, recall, and F1-score to determine the most effective model.

Table 1: Comparison of Performance Metrics of ML Algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score
RF	0.90	0.92	0.84	0.88
KNN	0.93	0.93	0.85	0.90
SVM	0.94	0.95	0.86	0.91
XGBoost	0.96	0.96	0.88	0.92
Conv XGBoost	**0.97**	**0.97**	**0.88**	**0.93**

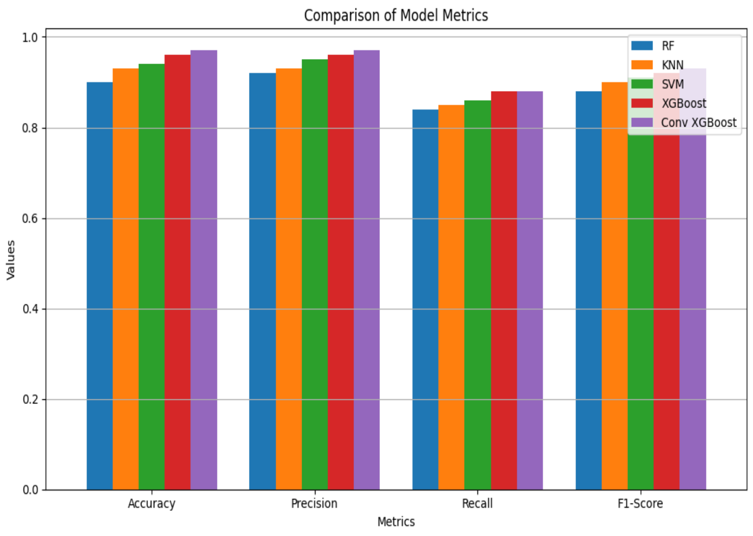


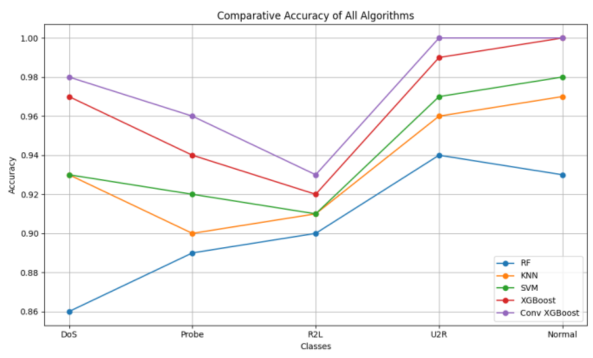
Fig.5. Comparison of ML Algorithms

The above table and figure contrasts the efficacy of five machine learning algorithms—RF, KNN, SVM, XGBoost, and Conv XGBoost—according to Accuracy, Precision, Recall, and F1-Score. Conv XGBoost exhibits the greatest values across all measures (accuracy 0.97, precision 0.97, recall 0.88, F1-score 0.93), succeeded by XGBoost (accuracy 0.96, precision 0.96, recall 0.88, F1-score 0.92). SVM and KNN demonstrate robust performance, but marginally inferior to XGBoost and Conv XGBoost, whilst RF exhibits the least performance overall.

Table 2: Comparison of ML Algorithm Performance Based on Accuracy and Precision

	Accuracy						**Precision**				
Class	**RF**	**KNN**	**SVM**	**XGBoost**	**Conv XGBoost**		**RF**	**KNN**	**SVM**	**XGBoost**	**Conv XGBoost**
DoS	0.86	0.93	0.93	0.97	0.98		0.93	0.97	0.98	0.98	0.99
Probe	0.89	0.90	0.92	0.94	0.96		0.93	0.95	0.96	0.96	0.97
R2L	0.90	0.91	0.91	0.92	0.93		0.92	0.92	0.94	0.96	0.98
U2R	0.94	0.96	0.97	0.99	1.00		0.83	0.85	0.89	0.89	0.90
Normal	0.93	0.97	0.98	1.00	1.00		0.97	0.97	0.98	0.99	1.00

The above table and figure contrasts the Accuracy and Precision of five algorithms (RF, KNN, SVM, XGBoost, and ConvXGBoost) across five categories. ConvXGBoost attains the best accuracy in most categories: DoS (0.98), Probe (0.96), R2L (0.93), U2R (1.00), and Normal (1.00), as well as the most precision in DoS (0.99), Probe (0.97), R2L (0.98), U2R (0.90), and Normal (1.00). XGBoost exhibits accuracy values of: DoS (0.97), Probe (0.94), R2L (0.92), U2R (0.99), and Normal (1.00), and precision values of: DoS (0.98), Probe (0.96), R2L (0.96), U2R (0.89), and Normal (0.99). SVM and KNN exhibit commendable performance, albeit marginally inferior, particularly in U2R. Random Forest exhibits the lowest overall performance, especially in U2R, with an accuracy of 0.94 and a precision of 0.83.



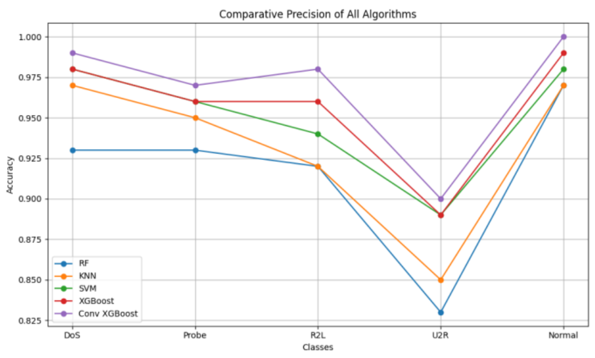
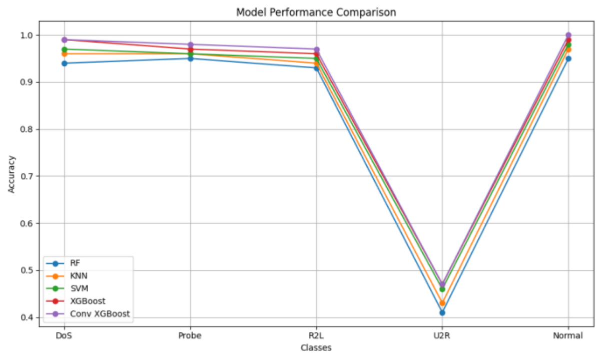


Fig.6. Comparative Analysis of ML algorithms based on Accuracy and Precision.

Table 3: Comparison of ML Algorithm Performance Based on Recall and F1-Score

	Recall						**F1-Score**				
Class	**RF**	**KNN**	**SVM**	**XGBoost**	**Conv XGBoost**		**RF**	**KNN**	**SVM**	**XGBoost**	**Conv XGBoost**
DoS	0.94	0.96	0.97	0.99	0.99		0.95	0.97	0.98	0.99	1.00
Probe	0.95	0.96	0.96	0.97	0.98		0.96	0.97	0.98	0.99	1.00
R2L	0.93	0.94	0.95	0.96	0.97		0.96	0.97	0.98	0.98	0.99
U2R	0.41	0.43	0.46	0.47	0.47		0.58	0.60	0.62	0.64	0.65
Normal	0.95	0.97	0.98	0.99	1.00		0.97	0.98	0.98	0.99	1.00



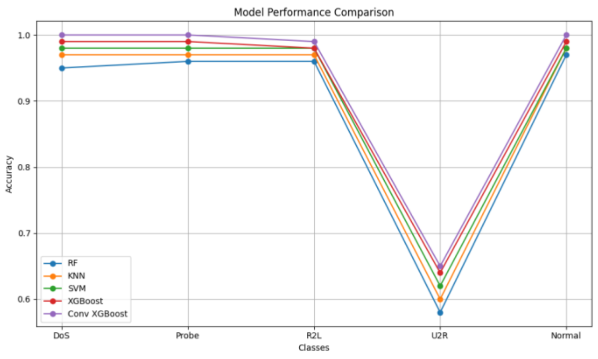


Fig.7. Comparative Analysis of ML Algorithms based on Recall and F1-Score.

The above table and figure juxtaposes the Recall and F1-Score of five algorithms (RF, KNN, SVM, XGBoost, and ConvXGBoost) across five categories. ConvXGBoost attains the highest recall and F1-score across most categories, exhibiting recall values of DoS (0.99), Probe (0.98), R2L (0.97), U2R (0.47), and Normal (1.00), alongside F1-scores of DoS (1.00), Probe (1.00), R2L (0.99), U2R (0.65), and Normal (1.00). XGBoost exhibits strong performance with recall values of DoS (0.99), Probe (0.97), R2L (0.96), U2R (0.47), and Normal (0.99), alongside F1-scores of DoS (0.99), Probe (0.99), R2L (0.98), U2R (0.64), and Normal (0.99). SVM and KNN provide robust performance; however they lag marginally behind XGBoost and Convolutional XGBoost. RF exhibits the lowest recall and F1-scores, particularly in U2R (recall 0.41, F1-score 0.58).

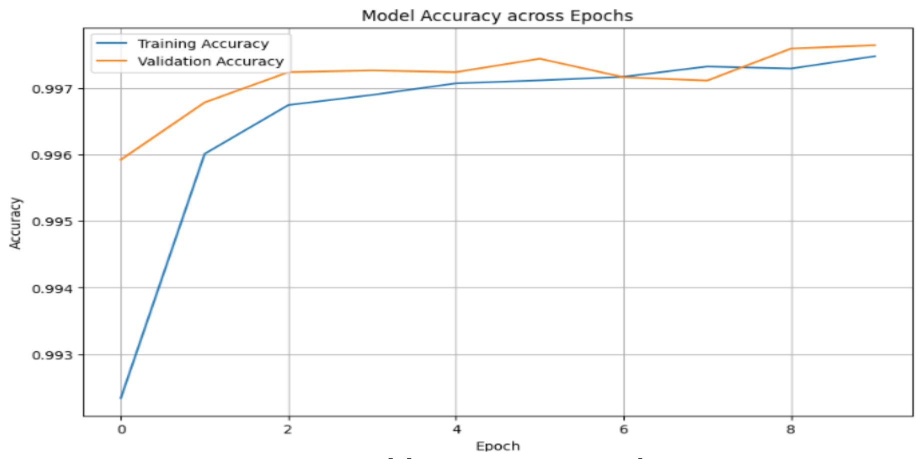


Fig.8. Model Accuracy over Epochs

The above training log illustrates the performance of a deep learning model throughout 10 epochs. The model commences with an accuracy of 99.23% and progressively enhances, attaining 99.75% by the concluding epoch. The validation accuracy rises from 99.59% to 99.76%, accompanied by a reduction in validation loss, signifying effective generalization. The test accuracy reaches 99.77%, validating the model's exceptional performance.

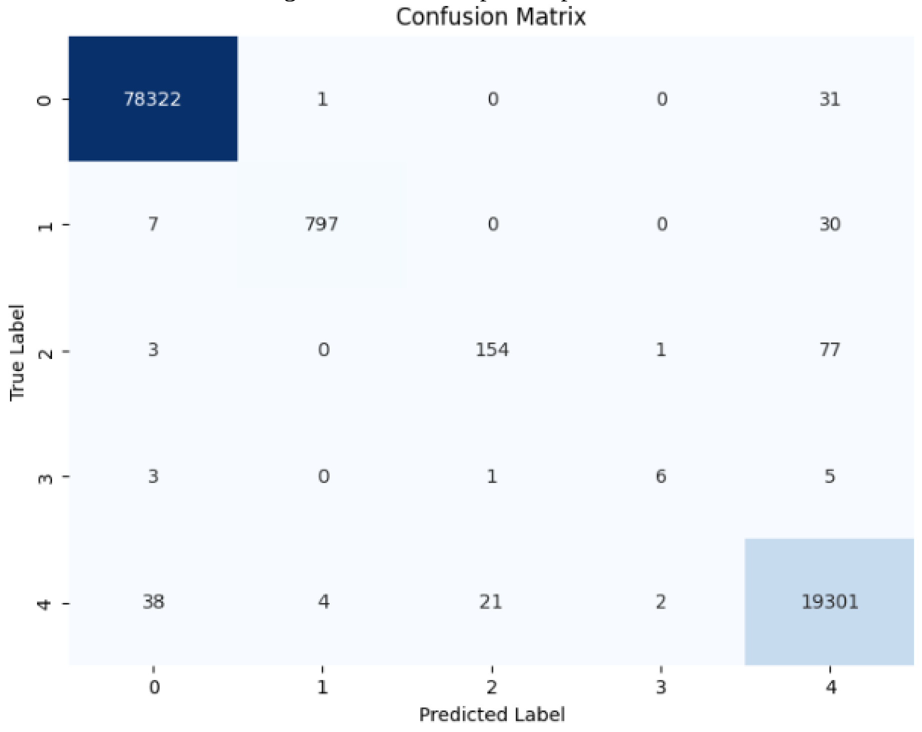


Fig.9. Confusion Matrix for Proposed Model

The above Confusion Matrix provides a visual assessment of a categorization model's efficacy. The confusion matrix presented above evaluates the performance of a classification model. The matrix consists of five classes (0 to 4) for both true and predicted labels. The highest number of correctly classified instances is in class 0, with 78,322 true positives, followed by class 4 with 19,301 true positives. Class 1 has 797 correctly classified instances, while class 2 and class 3 have 154 and 6 correctly classified instances, respectively. Misclassifications are observed across all classes, with notable cases such as 31 false positives in class 0, 30 in class 1, and 77 in class 2. Class 4 shows minor misclassification with 38 instances predicted as class 0, 4 as class 1, and 21 as class 2. Overall, the model demonstrates high accuracy for classes 0 and 4, while performance declines for class 3 due to fewer correct predictions.

5. CONCLUSION

The proposed research enhances intrusion detection by evaluating five machine learning algorithms such as RF, KNN, SVM, XGBoost, and ConvXGBoost across key performance metrics. The findings highlight ConvXGBoost as the most effective model, achieving the highest accuracy (0.97), precision (0.97), recall (0.88), and F1-score (0.93), demonstrating its superior ability to detect cyber threats. By integrating CNNs with XGBoost, the approach improves feature extraction and classification performance, reducing false positives and increasing detection accuracy. The experimental results, confusion matrix, and training analysis validate the model's efficiency in real-world cybersecurity applications. This research contributes to developing robust, scalable, and adaptive IDS solutions capable of mitigating evolving cyber threats effectively.

REFERENCES

1. S. M, A. G. S, H. B, A. S and M. G, "Intrusion Detection System Using Web Application," 2024 Ninth International Conference on Science Technology Engineering and Mathematics (IEEE), Chennai, India, 2024, pp. 1-6.
2. M. K. Nallakaruppan, S. R. K. Somayaji, S. Fuladi, F. Benedetto, S. K. Ulaganathan and G. Yenduri, "Enhancing Security of Host-Based Intrusion Detection Systems for the Internet of Things," in IEEE Access, vol. 12, pp. 31788-31797, 2024, doi: 10.1109/ACCESS.2024.3355794.
3. Aljuaid, W.H.; Alshamrani, S.S. A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments. Appl. Sci. 2024, 14, 5381.
4. Chua, T.-H.; Salam, I. Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection Using Progressive Dataset. Symmetry 2023, 15, 1251.
5. Awajan, A. A Novel Deep Learning-Based Intrusion Detection System for IoT Networks. Computers 2023, 12, 34.

6. V. Budania, M. Ahmed and A. Verma, "Deep Learning with Encoders for Intrusion Detection Systems (IDS)," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2023, pp. 1604-1609.

7. Sydney Mambwe Kasongo, A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework, Computer Communications, Volume 199, 2023, Pages 113-125, ISSN 0140-3664.

8. Rushendra, K. Ramli, N. Hayati, E. Ihsanto, T. S. Gunawan and A. H. Halbouni, "Development of Intrusion Detection System using Residual Feedforward Neural Network Algorithm," 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2021, pp. 539-543.

9. Cao, B.; Li, C.; Song, Y.; Qin, Y.; Chen, C. Network Intrusion Detection Model Based on CNN and GRU. Appl. Sci. 2022, 12, 4184.

10. Thirimanne, S.P., Jayawardana, L., Yasakethu, L. et al. Deep Neural Network Based Real-Time Intrusion Detection System. SN COMPUT. SCI. 3, 145 (2022).

11. Azar, A.T., Shehab, E., Mattar, A.M. et al. Deep Learning Based Hybrid Intrusion Detection Systems to Protect Satellite Networks. J Netw Syst Manage 31, 82 (2023).

12. Asgharzadeh, H., Ghaffari, A., Masdari, M. et al. An Intrusion Detection System on the Internet of Things Using Deep Learning and Multi-objective Enhanced Gorilla Troops Optimizer. J Bionic Eng 21, 2658–2684 (2024).

13. B. K. A and M. Vijayakumar, "Enhancing Intrusion Detection System (IDS) Through Deep Packet Inspection (DPI) with Machine Learning approaches," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-7.

14. Du, C.; Guo, Y.; Zhang, Y. A Deep Learning-Based Intrusion Detection Model Integrating Convolutional Neural Network and Vision Transformer for Network Traffic Attack in the Internet of Things. Electronics 2024, 13, 2685.

15. Mubarak Albarka Umar, Zhanfang Chen, Khaled Shuaib, Yan Liu, Effects of feature selection and normalization on network intrusion detection, Data Science and Management, Volume 8, Issue 1, 2025, Pages 23-39, ISSN 2666-7649.

16. B. Al-Fuhaidi, Z. Farae, F. Al-Fahaidy, G. Nagi, A. Ghallab, and A. Alameri, "Anomaly-Based Intrusion Detection System in Wireless Sensor Networks Using Machine Learning Algorithms," Appl. Comput. Intell. Soft Comput., vol. 2024, Art. no. 2625922, Sep. 2024.

17. Javed, A.; Ehtsham, A.; Jawad, M.; Awais, M.N.; Qureshi, A.-u.-H.; Larijani, H. Implementation of Lightweight Machine Learning-Based Intrusion Detection System on IoT Devices of Smart Homes. Future Internet 2024, 16, 200. https://doi.org/10.3390/fi16060200.

18. M. Benmalek and K.-D. Haouam, "Advancing Network Intrusion Detection Systems with Machine Learning Techniques," Adv. Artif. Intell. Mach. Learn., vol. 4, no. 3, pp. 2575–2592, Sep. 2024.