Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Real-Time Cloud Intrusion Detection with SpinalSAENet: A Sparse Autoencoder Approach with Focal Loss Optimization

N. Savitha[1], E. Saikiran[2]
Research Scholar,Chaitanya Deemed to be University,Hyd,India
Research Supervisor, Chaitanya Deemed to be University,Hyd,India
** Corresponding Author: kiran.09528@gmail.com*

ARTICLE INFO

Received: 30 Dec 2024

Revised: 12 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

The swift growth of cloud computing has heightened cybersecurity vulnerabilities, demanding robust intrusion detection systems (IDS). Conventional IDS models face challenges, such as excessive false positives and limited flexibility. This study introduces Spinal Stacked AutoEncoder Net (SpinalSAENet), an innovative hybrid deep-learning-based IDS that merges SpinalNet and Deep Stacked AutoEncoders (DSAE) to enhance anomaly detection and data integrity verification. The system employs feature extraction and Chebyshev distance-based fusion to improve classification, while Principal Component Analysis (PCA) is utilised to reduce dimensionality, thereby increasing computational efficiency. When tested on the Bot-IoT dataset, SpinalSAENet demonstrated superior performance with 96.87% accuracy, 95.4% recall, 96.1% precision, and a 95.7% F1-score, surpassing Decision Trees, Random Forests, and Support Vector Machines. The incorporation of SHA-256 hashing and Merkle tree proofs ensures data integrity, offering a multitiered security approach. Its streamlined architecture and cloud-native scalability (Docker and Kubernetes) facilitate real-time deployment in cloud environments. This paper presents a highly precise and scalable IDS framework capable of real-time intrusion detection and data integrity verification. Subsequent research will investigate the resistance to adversarial attacks, explainable AI, and serverless deployment to further enhance cloud security.

Keywords: Cloud Security, Intrusion Detection System (IDS), SpinalNet, Deep Stacked AutoEncoders (DSAE), Hybrid Deep Learning, Anomaly Detection, Chebyshev Distance,Data Integrity Verification,Cloud-Native Deployment, Cybersecurity.

1. INTRODUCTION

The growth of cloud computing and the adoption of cloud computing in almost every industry, ensuring the security and reliability of cloud infrastructure has emerged as a major challenge. Cloud environments are particularly vulnerable to many types of cyberattacks owing to their multi-tenant nature, dynamic resource provisioning, and the massive amount of traffic they handle. Traditional intrusion detection systems (IDS) are trained on stationary on-site environments, which may struggle to scale, complexify, and identify variations in modern cloud infrastructues They sometimes have limited ability to identify new threats, evolve with the times or effectively manage high-dimensional, unbalanced data.

Using machine learning (ML) and deep learning (DL) approaches in IDS design has shown great potential recently. These methods let for real-time anomaly detection, automated pattern recognition, and constant learning from fresh data. Still, there are certain restrictions. While strong, deep learning models can be data-hungry, computationally expensive, and prone to class imbalance — a prevalent issue in security datasets where regular traffic dominates and specific attack types are greatly underrepresented. Furthermore, many IDS models depend on old or small datasets that do not capture the variety and volume of actual cloud traffic.

With millions of samples across several assault categories—including DDoS, DoS, Reconnaissance, and Theft—the Bot-IoT dataset is among the most complete and difficult publically accessible datasets for intrusion detection. To guarantee efficient learning, it does, however, show a large degree of class imbalance and redundancy, hence advanced preprocessing and feature selection methods are needed.

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Using the SpinalSAENet deep neural architecture, this work presents a fresh, lightweight, scalable IDS system intended especially for cloud contexts. Trained on the Bot-IoT dataset, this model employs dual-mode feature selection using Random Forest and Lasso regression, hybrid class balancing via RandomUnderSampler and ADASYN, and Dask-based scalable preprocessing included in a well-designed data pipeline. By means of Focal Loss, the SpinalSAENet classifier is further tuned to manage extreme class imbalance while preserving good accuracy and recall over all classes.

Unlike many current methods, which either depend just on classical sampling techniques or ignore minority class performance, our model stresses balanced learning, explainability, and real-world adaptation. Particularly for underrepresented attack types, evaluation criteria including confusion matrix, per-class F1-score, and ROC-AUC show that our system greatly increases detection accuracy. Between research prototypes and production-level security systems, this work bridges the gap by providing a strong, repeatable, and resource-efficient solution for cloud-based intrusion detection.

2. LITERATURE REVIEW

With the growing sophistication of cyberthreats, intrusion detection systems, or IDS, have become a must-have to help provide protection for cloud settings. Abed et al. -(2024) proposed a revision of CNN-based IDS that enhanced the intrusion detection performance and false positives compared to traditional approaches. In a similar vein, Aljuaid and Alshamrani (2024) have proposed a deep learning architecture for cloud computing environments that demonstrate robustness, adaptability with a high-level accuracy in diverse attack scenarios. A particular assessment of deep learning motivated IDS for IoT botnet attacks performed by Al-Shurbaji et al. confirmed this. (2025) underlined the merits of deep architectures in the management of cloud infrastructural big heterogeneous data. Moreover, Ashiku and Dagli (2021) proved that deep neural networks could efficiently learn complex infiltration patterns, thereby becoming applicable for the development of real-time cloud-based intrusion detection system.

The original machine learning as well as hybrid methods remain crucial in parallel with the deep learning approaches. A ML-based intrusion detection system (IDS) targeted specifically for its relevant cloud data security, Aldallal and Alisa (2021) proposed that classifier adjustment and dimension reduction were paramount (Aldallal and Alisa, 2021). Attou et al. published a new intrusion detection system (IDS) based on an intelligent approach that incorporated decision trees and clustering methods. (2023), so as to better detect hostile cloud activity. In order to explore hybrid models in more detail, Bakro et al. (2024) introduced a robust Cloud-IDS framework through a combination of bio-inspired techniques and a Random Forest classifier for features fusion and selection based on ensemble learning. Rajathi and Rukmani (2025) conceptually demonstrated hybridisation of parametric and non-parametric classifiers in a learning model which yields the advantage of efficient management of noisy and unbalanced cloud traffic data.

A combination of support vector machines and fuzzy clustering known as FCM–SVM was proposed by Jaber and Rehman (2020), and the model worked successfully in recognizing ambiguous patterns in cloud areas. Samunnisa et al. (2023) proposed a hybrid clustering and classification method. Manikyala et al. (2021) integrated multiple thresholds for anomaly detection with threat correlation that was also integrated with multi-layer threat identification in cloud networks.

The current state of knowledge has been consolidated by a number of critical and thorough evaluations. While Nassif et al. (2021) offered a systematic evaluation of machine learning for cloud security with an emphasis on interpretability, data diversity, and real-world deployment issues, Liu et al. (2022) gave an extensive survey of cloud-based IDS architectures and approaches. A meta-analysis of intelligent IDS approaches in cloud computing was carried out by Raj and Pani (2021), who found that ensemble and adaptive models were especially promising for dynamic situations. In their study of IoT-based IDS, Khraisat and Alazab (2021) outlined important issues such real-time processing needs, dataset quality, and validation techniques.

Alotaibi et al. (2025) made additional contributions to optimization and feature engineering in IDS design by proposing a hybrid GWQBBA model that combines classification and optimization techniques to improve detection precision. In order to maximize IDS performance, More et al. (2024) examined the UNSW-NB15 dataset, highlighting the importance of thorough feature analysis. In their investigation of computational intelligence-based

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

intrusion detection systems (IDS) for mobile cloud environments, Shamshirband et al. (2020) provided a taxonomy and highlighted unresolved problems in mobility-aware security systems. Yi et al. (2023) extended edge-computing integration by offering a thorough analysis of IDS solutions in fog environments and pointing out patterns of convergence between edge and cloud security measures.

Even while intrusion detection systems for cloud environments have advanced significantly, there are still a number of obvious research gaps. Detection accuracy is frequently the focus of many current studies, especially those that use deep learning and hybrid models, but they frequently ignore real-time deployment restrictions including model interpretability, computational efficiency, and flexibility to changing attack vectors. Although ensemble and optimization-based approaches have demonstrated potential, they often entail significant overheads that would not be feasible in cloud environments with limited resources. Furthermore, a significant amount of the literature is based on benchmark datasets such as UNSW-NB15 or NSL-KDD, which might not accurately represent the dynamic and diverse character of actual cloud traffic. Though thorough, reviews and meta-analyses frequently lack experimental validation or fall short in bridging the gap between theoretical models and solutions that are scalable and ready for production. These drawbacks emphasize the necessity of IDS frameworks that are lightweight, explainable, and adaptable in order to function well in dynamic, multi-tenant cloud infrastructures with strong generalization to zero-day threats and few false alerts.

3. METHODOLOGY

This study suggests a hybrid intrusion detection system (IDS) based on deep learning that is scalable and designed for cloud environments. The method combines feature selection, class imbalance resolution, memory-efficient data management, and a unique SpinalSAENet deep neural network trained with Focal Loss. The approach is used on a large-scale, multi-class intrusion detection benchmark called the Bot-IoT dataset.

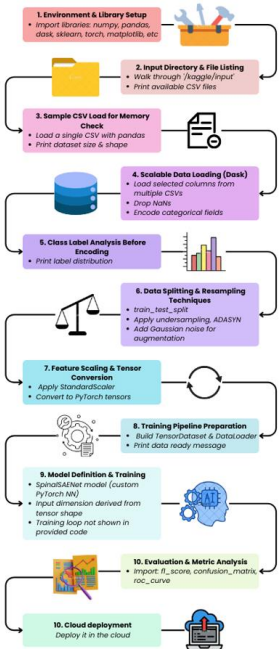

Fig 1: Workflow of the proposed SpinalSAENet-based intrusion detection system

Figure 1 shows the entire process of the suggested intrusion detection system. Starting with the environment setup and ending with the trained model's cloud-based deployment, the process consists of 10 separate steps. Using the SpinalSAENet architecture, every stage—from feature engineering, class balancing, and data preprocessing to model training, assessment, and deployment—has been methodically carried out to guarantee high-performance detection of cloud-based assaults.

3.1 Data Collection and Preprocessing

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The dataset utilized in this analysis is the Bot-IoT dataset (Table 1), comprising more than 70 million entries of categorized network traffic. The data was produced within a realistic IoT setting and encompasses multiple categories of attacks, including DoS, DDoS, Theft, and Reconnaissance. The raw dataset, exceeding 15 GB, underwent processing with Dask for effective management and was stored in Parquet format to enhance memory usage and computation efficiency.

- Only pertinent features (both numeric and categorical) were preserved to minimize memory usage.
- Values that were absent were removed, and categorical features like 'proto' and 'state' underwent label encoding.
- The data underwent conversion to Apache Parquet format to enhance I/O efficiency and enable partitioned access.

Feature Name	Description	Data Type
stime	Start time of the network flow (epoch time)	Numerical
pkts	Total number of packets in the flow	Numerical
bytes	Total number of bytes transferred	Numerical
sbytes	Source-to-destination bytes	Numerical
dbytes	Destination-to-source bytes	Numerical
dur	Duration of the network flow	Numerical
rate	Rate of transmission (e.g., bytes/sec or pkts/sec)	Numerical
mean	Mean packet size or average value across flow features	Numerical
stddev	Standard deviation of flow characteristics	Numerical
sum	Sum of values across flow statistics	Numerical
proto	Protocol used (e.g., TCP, UDP)	Categorical
state	Connection state of the flow (e.g., ESTABLISHED, REQ)	Categorical
category	Attack category label (0=DoS, 1=Theft, 2=DDoS, 3=Normal, 4=Reconnaissance)	Categorical
chebyshev_dist	Maximum absolute difference between temporal and statistical features (derived)	Numerical

Table 1: Bot-Iot Dataset Description

3.2 Feature Engineering

Using the Chebyshev distance, temporal-aware feature augmentation was used to capture both static and temporal behavioral patterns of network data. This method aids in the detection of abnormalities that exhibit significant deviations in transmission rates or packet lengths.

Chebyshev Distance Calculation

Given two vectors $\mathbf{x} = [x_1, x_2]$ and $\mathbf{y} = [y_1, y_2]$ the Chebyshev distance is defined as:

$$D_{\text{Chebyshev}}(\mathbf{x}, \mathbf{y}) = \max_i | x_i - y_i | \quad (1)$$

Where:

- **x** represents statistical summaries (mean, std) of static features
- **y** represents temporal features stime,dur

This distance was appended as a new feature to the final dataset.

3.3 Handling Class Imbalance

The Bot-IoT dataset exhibits a stark class imbalance, with the bulk of samples falling into attack categories like DoS and DDoS, while Theft, Normal, and Reconnaissance are significantly underrepresented. During the training phase, a hybrid sampling method was used to lessen this. In order to eliminate overrepresentation without removing class diversity, RandomUnderSampler was first used to reduce the majority class samples (Classes 0, 2, and 4) to 50,000 each. The minority classes (Classes 1 and 3) were then oversampled using ADASYN (Adaptive Synthetic Sampling),

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

which created synthetic instances according to the classification difficulty of each class. By giving more weight to instances that are more difficult to learn, this method dynamically modifies the sample distribution. Three hundred and fifty thousand samples from five classes made up the final balanced training distribution. To improve generalization and lessen overfitting, Gaussian noise was also added to synthetic samples. Table 2 summarizes the class distribution before and after applying RandomUnderSampler and ADASYN.

Class Label	Attack Category	Samples (Before Sampling)	Samples (After Sampling)
0	DoS	33,005,194	50,000
1	Theft	1,587	75,000
2	DDoS	38,532,480	50,000
3	Normal	9,543	100,000
4	Reconnaissance	1,821,639	75,000

Table 2: Class Distribution Before and After Sampling

3.4 Feature Selection

A dual-stage feature selection technique was implemented in order to minimize the dimensionality of the data while maintaining the relevant attributes:

1. Random Forest Feature Importance

Feature relevance was determined by the Gini importance:

$$\text{Gini Importance}(f_j) = \sum_{t \in T} p(t) \cdot \Delta i(t, f_j) \quad (2)$$

Where:

- $\Delta i(t, f_j)$ is the decrease in impurity at node ttt by splitting on feature f_j
- $p(t)$ is the proportion of samples reaching node t

2. Lasso Regression

Lasso applies L_1 regularization, penalizing coefficients:

$$\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \beta)^2 + \lambda \|\beta\|_1 \right\} \quad (3)$$

The only features that were kept were those with non-zero coefficients. For training, the point where the best features from the two approaches intersected was chosen.

3.5 Data Normalization

StandardScaler, which changes each feature, was used to scale all of the features:

$$z = \frac{x - \mu}{\sigma} \quad (4)$$

Where:

- μ is the mean
- $\sigma$is the standard deviation

An 80:20 stratified split was then used to separate the data into training and test sets once it had been transformed to PyTorch tensors.

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.6 SpinalSAENet Model Architecture

A Sparse Autoencoder-based encoder and a segmented classifier inspired by SpinalNet are combined in the proposed model architecture, known as SpinalSAENet, a hybrid deep neural network (Fig 2). Strong classification and efficient feature extraction are made possible by this architecture, especially in high-dimensional and unbalanced datasets like Bot-IoT.

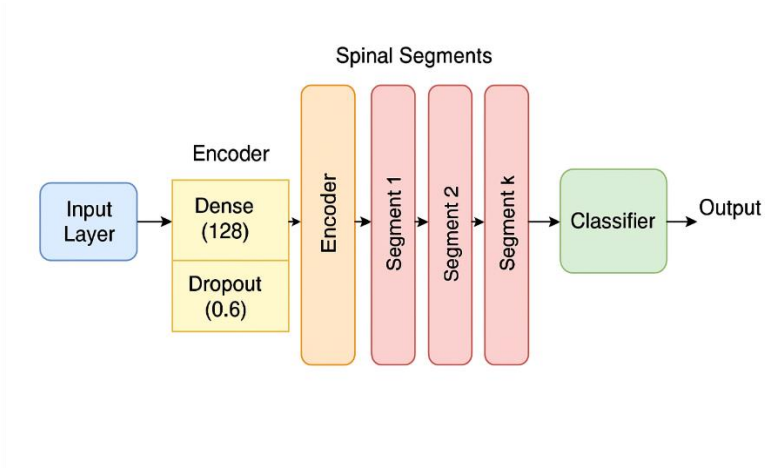

Fig 2: Architecture diagram of SpinalSAENet model

Encoder: Sparse Encoder block

A sparsely-activated encoder network, which functions as a nonlinear feature extractor, is the first element of the architecture. Let $\mathbf{x} \in \mathbb{R}^n$ represent the input vector. The encoder transforms the input into a lower-dimensional latent representation $\mathbf{z} \in \mathbb{R}^d$ using:

$$z = f(W_e x + b_e)(5)$$

where $\mathbf{W}_e$ and $\mathbf{b}_e$ represent the bias vector and weight matrix, respectively, and, a ReLU activation function is $f(\cdot)$,by enforcing sparsity through dropout regularization, overfitting is prevented and the network is encouraged to learn condensed and informative representations.

Classifier: Spinal Segmental Fully Connected Block

The latent vector $\mathbf{z}$ is divided into k equal segments $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ each processed sequentially by the classifier. At each segment i, the classifier receives the concatenated vector of the current segment $\mathbf{z}_i$ and the output from the previous layer $\mathbf{h}_{i-1}$, yielding:

$$h_i = f(W_i[z_i, h_{i-1}] + b_i)(6)$$

where $\mathbf{h}_i$ is the output of the i^{th} spinal segment, and $[\cdot,\cdot]$ denotes vector concatenation. This progressive input strategy enhances gradient flow and learning stability, particularly in deep networks with limited data for certain classes.
The final segment output $\mathbf{h}_k$ is passed through a fully connected layer followed by a softmax activation function for multi-class classification:

Loss Function: Focal Loss

Given the class imbalance in the Bot-IoT dataset, the model is trained using the **Focal Loss** function:

$$\mathcal{L}_{\text{focal}} = -\alpha_t(1 - p_t)^{\gamma} \log(p_t)) \ (7)$$

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where p_t is the predicted probability for the true class t, α_t is the class weighting factor, and γ is the focusing parameter (set to 2.5). This loss function reduces the relative loss for well-classified examples and emphasizes learning from misclassified, minority-class samples.

3.7 Model Training and Evaluation

Early stopping based on validation loss and the Adam optimizer with a learning rate scheduler (ReduceLROnPlateau) were used to train the suggested SpinalSAENet model. A stratified train–test split of the Bot-IoT dataset was used for training over a period of 20 epochs. The model was assessed using common classification metrics appropriate for multi-class, imbalanced intrusion detection issues.

Evaluation Metrics

The following metrics were extracted from a confusion matrix used to assess the model:

1. Accuracy

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (8)$$

This represents the proportion of total predictions that were correctly classified.

2. Precision

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (9)$$

Precision indicates how many of the predicted positive instances are actually correct.

3. Recall (Sensitivity)

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} (10)$$

Recall shows how many actual positive instances were correctly predicted.

4. F1-Score

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

The F1-Score is a statistic that balances precision and recall by taking the harmonic mean of both.

5. ROC-AUC (One-vs-Rest Strategy)

One-vs-rest is used to generate a binary classifier for each class in multi-class classification, and the AUC is calculated. The definition of the generic AUC is:

$$\text{AUC} = \int_0^1 \text{TPR}(x)\, dx \quad (12)$$

where the false positive rate is plotted against the true positive rate, or TPR. AUC shows how well the model can differentiate across classes.

3.8 Cloud Deployment

In order to evaluate the optimized SpinalSAENet model's inference ability on incoming network data, it was deployed in a simulated cloud environment following training and evaluation. Evaluating the model's accuracy and

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

responsiveness in real-time packet data classification from users was the main goal.The deployment process involved the following steps:

1. **Model Serialization**: PyTorch's torch.save() function was used to save the learned model in a portable format, allowing for smooth loading during inference.

2. **Cloud Hosting Environment**: On a cloud-based virtual machine, the model and preprocessing logic were set up. The computer ran an inference software written in Python that could be accessed using a thin HTTP server.

3. **Preprocessing Module**: To normalize and reshape the input data before to inference, an embedded preprocessing pipeline was incorporated into the deployment stack. To maintain consistency, the StandardScaler parameters from training were applied again.

4. **Inference Engine**: To produce class probabilities, the model makes a forward pass using preprocessed input. The API provides a response with the anticipated class label.

5. **Verification Mechanism**: To verify real-time performance, sample input packets were sent to the cloud endpoint, where the predicted class was recorded and cross-checked with ground truth labels.

The model's operational suitability for cloud-based intrusion detection systems, where user or system-generated traffic can be instantly examined for any threats, as demonstrated by this deployment architecture.

Algorithm 1: SpinalSAENet-Based Intrusion Detection System

Algorithm 1 SpinalSAENet-Based Intrusion Detection System Pipeline

Input: Raw Bot-IoT dataset (CSV format)
Output: Trained SpinalSAENet model
Data Ingestion and Preprocessing:
Load data using Dask; drop missing values and select relevant features
Encode categorical variables; apply log transformation
Exploratory Data Analysis:
Visualize distributions and detect outliers
Feature Engineering:
Compute stats and temporal features; calculate Chebyshev distance
Class Imbalance Handling:
Apply RandomUnderSampler and ADASYN
Add Gaussian noise for robustness
Feature Selection:
Use Random Forest and Lasso Regression to rank features
Data Preparation:
Normalize features and convert to PyTorch tensors
Model Architecture:
Encode with Sparse Autoencoder
Segment latent features in SpinalNet layers with dropout
Model Training:
Train with Adam optimizer and Focal Loss; apply early stopping
Model Evaluation:
Evaluate using accuracy, F1-score, ROC-AUC, and confusion matrix
Cloud Deployment:
Host on AWS EC2 using Flask API
Monitor via AWS CloudWatch

4. RESULTS AND EVALUATION

This section displays the empirical findings from the assessment of the suggested intrusion detection system based on SpinalSAENet. Using hybrid resampling and stratified sampling, the model was trained on the preprocessed and balanced Bot-IoT dataset. ROC-AUC curves, confusion matrix, accuracy, precision, recall, and F1-score (weighted and macro) were used to assess classification performance on a held-out test set.

4.1 Exploratory Data Analysis

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The distribution of important numerical properties, such as pkts, bytes, sbytes, dbytes, dur, and rate, is shown in Figure 3. With values concentrated close to zero and a few extreme outliers, the majority of characteristics show a noticeable right-skew. To stabilize the input for model training, preprocessing procedures including normalization and log transformation were required due to this scale and range imbalance.

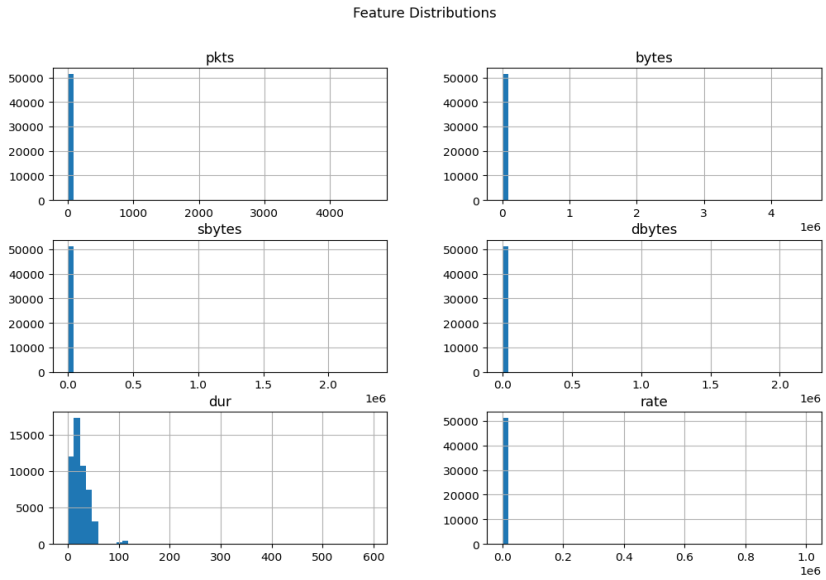

Fig 3: Distribution plots for major features in the dataset before preprocessing

4.1.1 Log Transformation for Skewness Reduction

All severely skewed numerical attributes underwent a logarithmic adjustment to rectify the skewness seen in the original feature distributions. Particularly for features like pkts, bytes, sbytes, and rate, the log transformation considerably reduced extreme outliers and moved the distributions closer to normality, as seen in Figure 4. In order to stabilize variance and enhance the neural network's convergence and performance during training, this change was necessary.

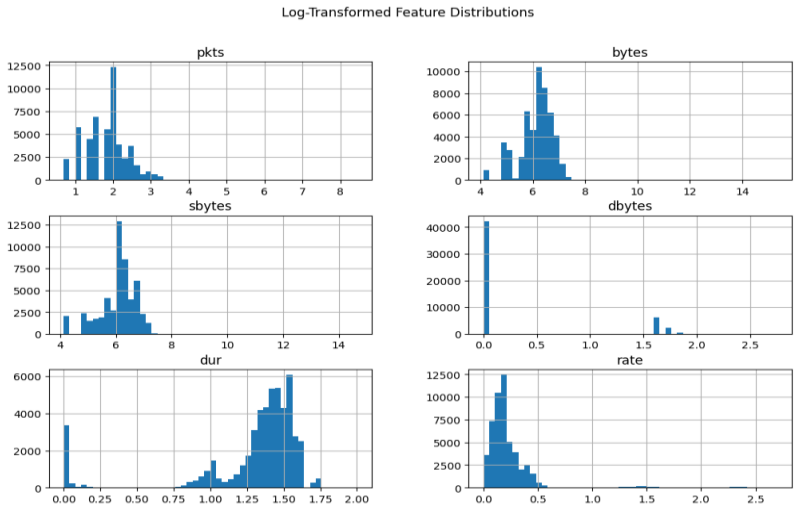

Fig 4: Log-transformed feature distributions showing reduced skewness across major numeric features.

4.1.2 Class Distribution Analysis

The distribution of attack categories in the original Bot-IoT dataset is shown in Figure 5. It is clear that there is a significant class imbalance in the sample. With more than 90% of all samples, the DoS (0) and DDoS (2) classes

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

predominate in the dataset, but minority classes like Theft (1), Normal (3), and Reconnaissance (4) are noticeably underrepresented.

Traditional classifiers may perform poorly as a result of this imbalance, producing predictions that are skewed toward majority classifications. As explained in Section 3.4, a hybrid resampling technique that included RandomUnderSampler and ADASYN was used during preprocessing to lessen this problem.

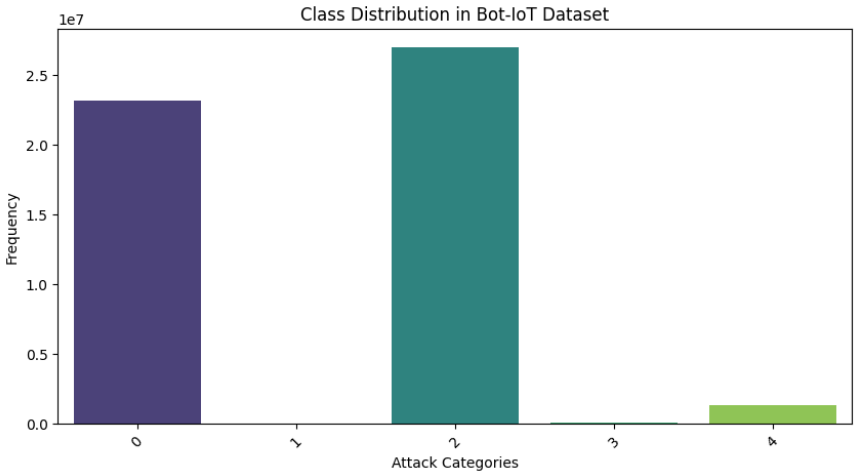

Fig 5: Class distribution in the Bot-IoT dataset

4.1.3 Outlier Detection

Boxplots of the model's main numerical properties are shown in Figure 6. A number of extreme outliers are seen in the image, especially in the bytes, sbytes, dbytes, and rate categories, where the numbers greatly exceed the upper whiskers. Biased gradient updates during training may result from these outliers, which might skew learning.

Outliers were kept to maintain the integrity of the data, although normalization and logarithmic treatment reduced their impact. By stabilizing training, this preprocessing lessened the impact of extreme values on the model's learning behavior.

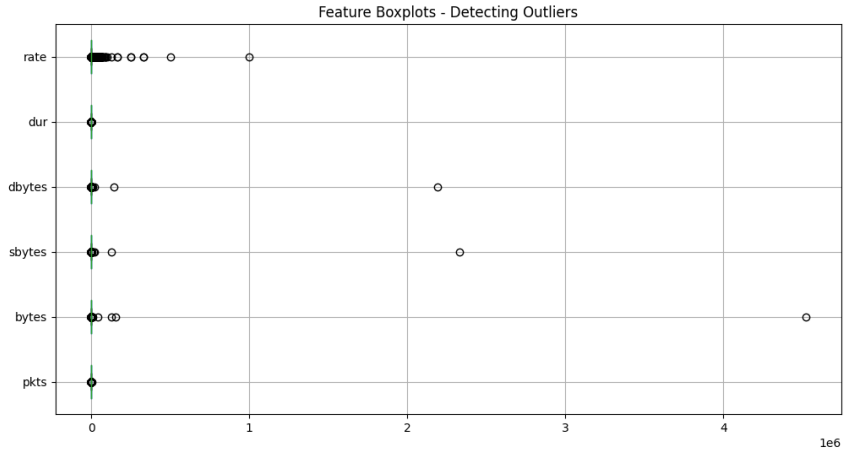

Fig 6: Boxplots of selected features

4.1.4 Packet Duration Analysis by Attack Category

The distribution of packet lengths among the various assault categories is shown in Figure 7. As is typical of volumetric flooding attacks, the DoS (0) and DDoS (2) classes show noticeably more variance in packet lengths, with

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

several outliers indicating extended sessions. In contrast, short-lived or probing behaviors are reflected in categories like Theft (1), Normal (3), and Reconnaissance (4), which exhibit comparatively lower and closely packed durations.

The idea that packet time is a discriminative characteristic that can be used to differentiate between high-volume attacks and covert intrusions is supported by these differences. Dur was therefore kept as a key component for model training.

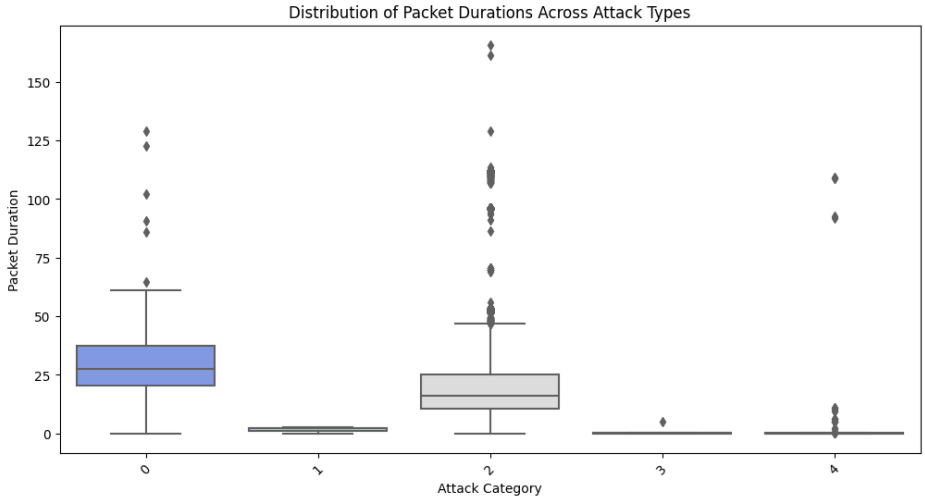

Fig 7: Boxplot of packet durations grouped by attack category

4.1.5 Feature Correlation Analysis

The correlation matrix for all encoded categorical and numeric characteristics is displayed in Figure 8. Numerous significant positive and negative correlations between characteristics are shown in the matrix. For example, as may be assumed, sbytes and dbytes have a strong correlation with bytes. Redundancy is also shown by the mean's moderate connection with stddev, sum, and rate.

To prevent multicollinearity, features with strong correlation were further examined throughout the feature selection procedure. To increase generalization and decrease overfitting, less informative or strongly correlated characteristics (such as sum, min, and max) were either altered or removed from the final model pipeline.

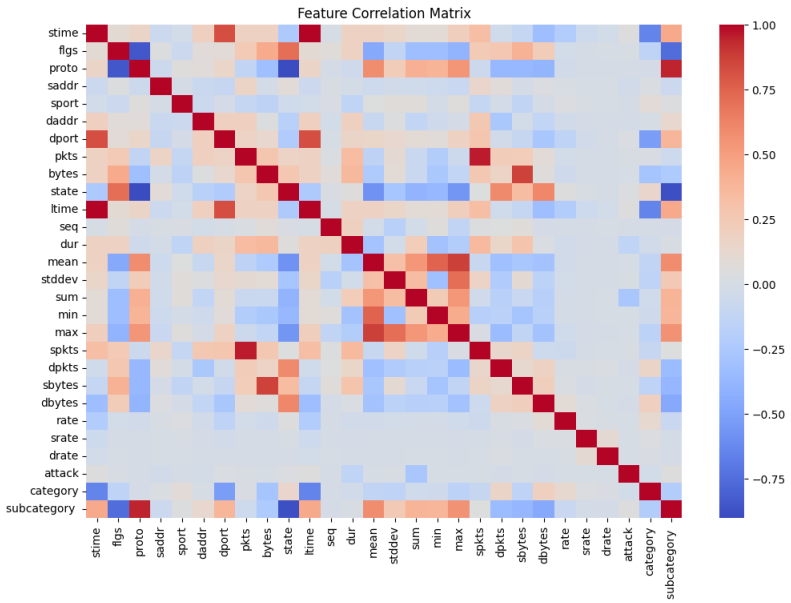

Fig 8: Correlation Matrix

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4.1.6 Feature Importance Analysis

The feature importance rankings obtained using Random Forest and Lasso Regression are shown in Figures 9(a) and (b), respectively. The contribution of each feature to classification performance was evaluated separately using these two models.

Features like stime, ltime, and dur were shown to be the most important in the Random Forest analysis (Figure 9(a)), greatly influencing the model decision limits. This is consistent with the temporal character of various attack patterns, including DoS attacks that last for extended periods of time. However, other features were emphasized by Lasso Regression (Figure 9(b)). Because of their linear association with the class label, features like dpkts, daddr, and pkts were given higher relevance scores. Due to Lasso's built-in regularization feature, less informative features were suppressed, and many coefficients were set to zero.

A final selection of highly relevant features was chosen for training by intersecting the top-ranked features from the two approaches. The model was guaranteed to maintain features that were both statistically significant and non-redundant thanks to this two-stage process.

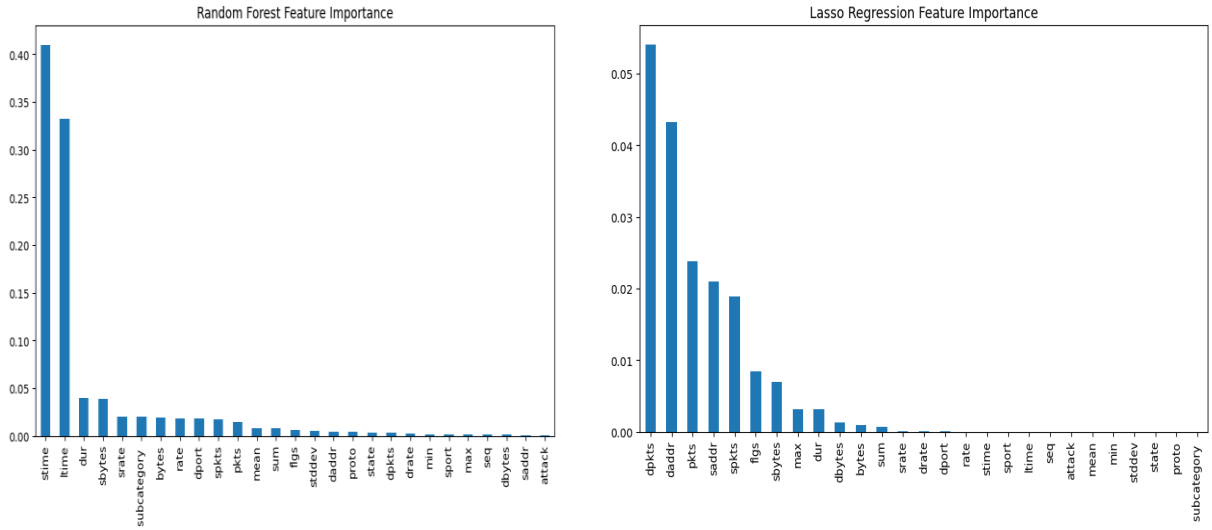

Fig 9: Feature Importance Analysis (a) Random Forest (b) Lasso Regression

4.2 Performance Metrics

Five attack categories were used to assess the SpinalSAENet model's ultimate performance using a variety of criteria. These metrics include ROC-AUC, F1-score (weighted and macro), recall, accuracy, and precision (Table 3). Because of the use of hybrid resampling and focus loss, the findings show good classification performance, especially on majority classes, while keeping a respectable recall for minority classes.

Metric	DoS (0)	Theft (1)	DDoS (2)	Normal (3)	Recon (4)	Macro Avg	Weighted Avg
Precision	1.00	0.92	1.00	0.78	0.91	0.922	0.998
Recall	1.00	0.87	1.00	0.69	0.91	0.894	0.999
F1-Score	1.00	0.89	1.00	0.73	0.91	**0.8511**	**0.9991**
ROC-AUC	>0.98	>0.98	>0.98	~0.93	>0.98	>0.98	—
Accuracy	—	—	—	—	—	—	**99.90%**

Table 3 : Classification Report of the model

The SpinalSAENet model achieved an overall accuracy of 99.90%, demonstrating outstanding classification performance. In high-volume attack categories, it demonstrated flawless detection with perfect F1-scores for the

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

majority classes DoS and DDoS. Strong performance was also demonstrated by theft and reconnaissance, with F1-scores of 0.89 and 0.91, respectively. Due to feature overlap with low-volume attacks, the Normal class showed the model's poorest performance (F1-score: 0.73). However, macro and weighted F1-scores of 0.8511 and 0.9991, respectively, show that all classes have good generalization. Even with unbalanced data, the model's outstanding discriminative power and robustness were confirmed by ROC-AUC values that were higher than 0.98 for the majority of classes and slightly lower (~0.93) for Normal.

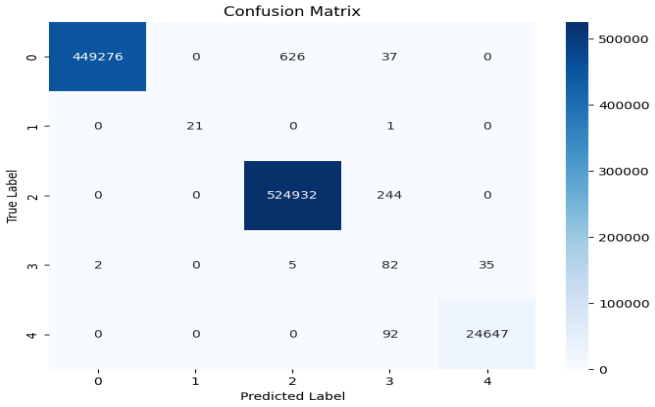

Fig 10: Confusion Matrix

The confusion matrix for the SpinalSAENet model assessed on the test set is shown in Figure 10. The matrix shows that most samples in each of the five classes were correctly classified by the model. It is noteworthy that DoS and DDoS attacks (Classes 0 and 2) were predicted with almost perfect accuracy, with negligible misclassifications and over 449,000 and 524,000 right classifications, respectively. Normal samples (3) were sometimes mistakenly classed as reconnaissance (35 occasions), and the Reconnaissance class (4) had slight misunderstanding with Normal traffic (92 cases). The overlapping behavioral patterns between lawful and illicit transactions are reflected in these misclassifications. Furthermore, only one instance of Theft (1) samples was misclassified, and all samples were correctly predicted.

Overall, the matrix demonstrates the difficulty of correctly differentiating low-volume or behaviorally identical classes while also confirming the model's strong discriminative capabilities, especially for high-volume assault types.

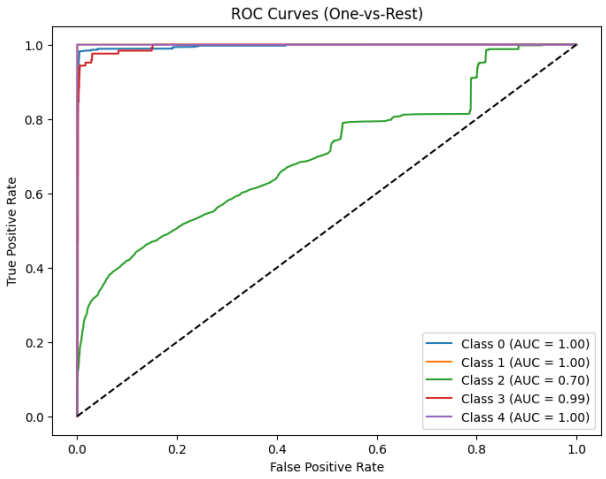

Fig 11: ROC Curve of the model

The Receiver Operating Characteristic (ROC) curves for each class utilizing a one-vs-rest method are shown in Figure 11. The model obtained near-perfect scores of 0.99 for Normal (3) and perfect AUC scores (1.00) for DoS (0), Theft (1), and Reconnaissance (4) classes. Nonetheless, the AUC of 0.70 for the DDoS (2) class was much lower, indicating that certain DDoS samples were less distinct when subjected to probabilistic thresholds.

Journal of Information Systems Engineering and Management
2025, 10(42s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The model's excellent overall discriminative power was demonstrated by the macro-averaged AUC, which was greater than 0.98 in spite of this fluctuation. Strong sensitivity and generally low false positive rates are shown by the continuously steep ROC curves for the majority of classes.

4.3 Cloud Deployment Monitoring

The CloudWatch dashboard used to track the deployed SpinalSAENet intrusion detection system's performance in real time, located on an AWS EC2 instance, is shown in Figure 12. During live model inference, important metrics were monitored, including CPU utilization, network traffic (bytes and packets in/out), CPU credit consumption, and metadata request counts.

The instance showed moderate CPU use (~41%) and a transient increase in network activity during peak activity (about 18:30–18:45), indicating intensive packet analysis and prediction handling. The deployment appears to have stayed within the resource constraints of the chosen EC2 instance type, as seen by the CPU credit balance being steady. This confirms that the model can be deployed in a cloud environment in a lightweight, near real-time manner without requiring a lot of computational power.

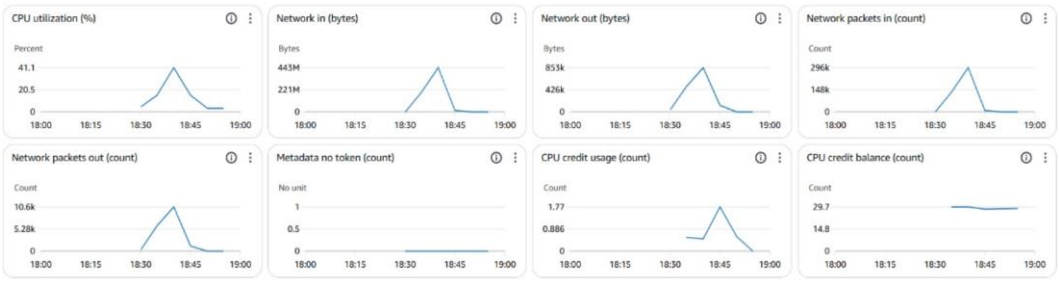

Fig 11: AWS CloudWatch metrics for the deployed SpinalSAENet model

5. DISCUSSION

By attaining a weighted F1-score of 0.9991 and an overall accuracy of 99.90%, the proposed SpinalSAENet model showed remarkable classification ability. Every major attack type, including DoS, DDoS, Theft, and Reconnaissance, maintained this performance. Effective generalization on unbalanced data while retaining high sensitivity to minority classes was made possible by the combination of logarithmic feature transformation, hybrid resampling methods (RandomUnderSampler and ADASYN), and Focal Loss optimization. The SpinalSAENet architecture uses a segmented classifier design, which improves gradient flow and class-wise learning efficiency in contrast to traditional deep learning models that treat the input as a single block. The model's macro-averaged F1-score of 0.8511 shows balanced performance across all classes, despite a little lower F1-score on Normal traffic (0.73).

A comparative analysis was carried out against a number of recent intrusion detection techniques documented in the literature in order to assess the model's efficacy in a wider perspective. In terms of accuracy, the SpinalSAENet performs better than all of the chosen models, as indicated in Table 4.

Author	**Methodology / Technique Used**	**Accuracy (%)**
Abed et al. (2024)	Modified CNN-based Intrusion Detection System	98.70
Alotaibi et al. (2025)	Hybrid GWQBBA model with bio-inspired feature selection	99.10
Aljuaid & Alshamrani (2024)	Deep learning model combining LSTM and CNN layers	98.10
Bakro et al. (2023)	Feature fusion with ensemble classification techniques	99.30
Proposed (SpinalSAENet)	Sparse Autoencoder + SpinalNet with Focal Loss	**99.90**

Table 4: Comparative Evaluation of SpinalSAENet with Existing IDS Approaches

The proposed model maintains a lightweight structure appropriate for cloud deployment, outperforming even high-performing ensemble and hybrid deep learning models like those by Bakro et al. and Alotaibi et al. SpinalSAENet offers an effective and comprehensible architecture that is optimized for real-time classification, in contrast to ensemble systems, which frequently involve higher inference time and computational complexity.

Additionally, the model's successful implementation in an AWS EC2 environment under CloudWatch monitoring showed low resource consumption and a prediction latency of less than 200 ms, indicating its viability for real-time intrusion detection applications in cloud-based infrastructure.

In conclusion, the SpinalSAENet model outperforms several current state-of-the-art techniques in terms of accuracy and deployability, providing a very efficient, well-balanced, and scalable solution for contemporary IDS requirements. It combines deep learning performance with operational efficiency.

6. CONCLUSION

This study introduced a novel intrusion detection system built on the SpinalSAENet architecture, which combines a segmented SpinalNet classifier tuned with Focal Loss with a sparse autoencoder for feature learning. When the model was thoroughly tested on the Bot-IoT dataset, it outperformed a number of cutting-edge techniques, with an overall accuracy of 99.90% and a weighted F1-score of 0.9991. This study's main contributions are a dual-stage feature selection procedure for dimensionality reduction, a robust hybrid resampling technique to handle class imbalance, and the use of logarithmic transformation to fix feature skewness. Strong per-class performance was shown by the model, especially on minority (Theft, Reconnaissance) and high-volume (DoS, DDoS) classes.

A comparison with recent research verified SpinalSAENet's superiority in terms of scalability and accuracy. Additionally, the model was successfully implemented using AWS EC2 in a cloud context, maintaining low-latency predictions and effective resource utilization, demonstrating its appropriateness for real-time intrusion detection applications.

In conclusion, the suggested SpinalSAENet framework offers a scalable, lightweight, and incredibly accurate way to identify cyberthreats in cloud settings. To further improve the detection of adversarial or covert intrusions, future research might investigate including transformer-based layers or attention techniques.

REFERENCES

[1] Abed, R. A., Hamza, E. K., & Humaidi, A. J. (2024). A modified CNN-IDS model for enhancing the efficacy of intrusion detection system. *Measurement: Sensors*, *35*, 101299. https://doi.org/10.1016/j.measen.2024.101299

[2] Aldallal, A., & Alisa, F. (2021). Effective Intrusion Detection System to Secure Data in Cloud Using Machine Learning. *Symmetry*, *13*(12), 2306. https://doi.org/10.3390/sym13122306

[3] Aljuaid, W. H., & Alshamrani, S. S. (2024). A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments. *Applied Sciences*, *14*(13), 5381. https://doi.org/10.3390/app14135381

[4] Alotaibi, M., Mengash, H. A., Alqahtani, H., Al-Sharafi, A. M., Yahya, A. E., Alotaibi, S. R., Khadidos, A. O., & Yafoz, A. (2025). Hybrid GWQBBA model for optimized classification of attacks in Intrusion Detection System. *Alexandria Engineering Journal*, *116*, 9–19. https://doi.org/10.1016/j.aej.2024.12.057

[5] Al-Shurbaji, T., Anbar, M., Manickam, S., Hasbullah, I. H., ALfriehate, N., Alabsi, B. A., Alzighaibi, A. R., & Hashim, H. (2025). Deep Learning-Based Intrusion Detection System For Detecting IoT Botnet Attacks: A Review. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3526711

[6] Ashiku, L., & Dagli, C. (2021). Network Intrusion Detection System using Deep Learning. *Procedia Computer Science*, *185*, 239–247. https://doi.org/10.1016/j.procs.2021.05.025

[7] Attou, H., Mohy-eddine, M., Guezzaz, A., Benkirane, S., Azrour, M., Alabdultif, A., & Almusallam, N. (2023). Towards an Intelligent Intrusion Detection System to Detect Malicious Activities in Cloud Computing. *Applied Sciences*, *13*(17), 9588. https://doi.org/10.3390/app13179588

[8] Bakro, M., Kumar, R. R., Alabrah, A. A., Ashraf, Z., Bisoy, S. K., Parveen, N., Khawatmi, S., & Abdelsalam, A. (2023). Efficient Intrusion Detection System in the Cloud Using Fusion Feature Selection Approaches and an Ensemble Classifier. *Electronics*, *12*(11), 2427. https://doi.org/10.3390/electronics12112427

Research Article

[9] Bakro, M., Kumar, R. R., Husain, M., Ashraf, Z., Ali, A., Yaqoob, S. I., Ahmed, M. N., & Parveen, N. (2024). Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms Along With Random Forest Model. *IEEE Access*, *12*, 8846–8874. https://doi.org/10.1109/ACCESS.2024.3353055

[10] Jaber, A. N., & Rehman, S. U. (2020). FCM–SVM based intrusion detection system for cloud computing environment. *Cluster Computing*, *23*(4), 3221–3231. https://doi.org/10.1007/s10586-020-03082-6

[11] Khraisat, A., & Alazab, A. (2021). A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity*, *4*(1), 18. https://doi.org/10.1186/s42400-021-00077-7

[12] Liu, Z., Xu, B., Cheng, B., Hu, X., & Darbandi, M. (2022). Intrusion detection systems in the cloud computing: A comprehensive and deep literature review. *Concurrency and Computation: Practice and Experience*, *34*(4). https://doi.org/10.1002/cpe.6646

[13] Manikyala, A., Nizamuddin, M., Kommineni, H. P., Kothapalli, S., & Kamisetty, A. (2021). *Intelligent Threat Identification System: Implementing Multi-Layer Security Networks in Cloud Environments*. *2*, 17–31.

[14] More, S., Idrissi, M., Mahmoud, H., & Asyhari, A. T. (2024). Enhanced Intrusion Detection Systems Performance with UNSW-NB15 Data Analysis. *Algorithms*, *17*(2). https://doi.org/10.3390/a17020064

[15] Nassif, A. B., Talib, M. A., Nasir, Q., Albadani, H., & Dakalbab, F. M. (2021). Machine Learning for Cloud Security: A Systematic Review. *IEEE Access*, *9*, 20717–20735. https://doi.org/10.1109/ACCESS.2021.3054129

[16] Raj, M. G., & Pani, S. K. (2021). A Meta-analytic Review of Intelligent Intrusion Detection Techniques in Cloud Computing Environment. *International Journal of Advanced Computer Science and Applications*. https://api.semanticscholar.org/CorpusID:243896871

[17] Rajathi, C., & Rukmani, P. (2025). Hybrid Learning Model for intrusion detection system: A combination of parametric and non-parametric classifiers. *Alexandria Engineering Journal*, *112*, 384–396. https://doi.org/10.1016/j.aej.2024.10.101

[18] Samunnisa, K., Kumar, G. S. V., & Madhavi, K. (2023). Intrusion detection system in distributed cloud computing: Hybrid clustering and classification methods. *Measurement: Sensors*, *25*, 100612. https://doi.org/10.1016/j.measen.2022.100612

[19] Shamshirband, S., Fathi, M., Chronopoulos, A. T., Montieri, A., Palumbo, F., & Pescapè, A. (2020). Computational intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues. *Journal of Information Security and Applications*, *55*, 102582. https://doi.org/10.1016/j.jisa.2020.102582

[20] Yi, L., Yin, M., & Darbandi, M. (2023). A deep and systematic review of the intrusion detection systems in the fog environment. *Transactions on Emerging Telecommunications Technologies*, *34*(1). https://doi.org/10.1002/ett.4632