# A Lightweight Gradient-Boosted Tree Framework for Intrusion Detection in Cloud Computing Environments

1st Muhammad Zameer Malik
*dept. of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, pakistan
zameermalik0348a@gmail.com

2nd Shaheer Ahmad
*dept. of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, pakistan
shaheermujaddadi2@gmail.com

3rd Aoun Muhammad
*dept. of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, pakistan
aoun.muhammad@iub.edu.pk

4th Sana Tariq
*dept. of Information and Communication Engineering*
*The Islamia University of Bahawalpur*
Bahawalpur, pakistan
Sana.tariq@iub.edu.pk

***Abstract*—Cloud computing infrastructures are increasingly targeted by sophisticated and continuously evolving cyberattacks, making robust Intrusion Detection Systems (IDS) essential. A large amount of recent work follows detection accuracy through deep learning or metaheuristic optimization. These methods are, however, very compute-intensive and cannot be practically used in resource-poor cloud nodes. In this paper we revisit a deliberately simple question: how much loss of accuracy is there if a single object is shown that is merely the wrong size, and how much loss of accuracy is there if both objects are shown and both are the wrong size? Faster and stronger than the heavier ensemble and hybrid model.Better than the heavier ensemble and hybrid model — well-configured gradient-boosted tree. designs?We propose a lightweight IDS framework based on an Extreme Gradient Boosting (XGBoost) classifier, and compare it with a baseline Random Forest, a LightGBM model, and a three-member soft-voting ensemble (Random Forest + Logistic Regression + Histogram Gradient Boosting). All models are assessed on the NSL-KDD benchmark and independently validated on the modern CICIDS2017 dataset, using ten random seeds per model with reported means, standard deviations, and paired significance tests. We additionally report training time, inference time, memory footprint, and CPU time, together with an ablation study and per-attack-class analysis. The results show that the gradient-boosted tree attains the highest accuracy and F1-score on both datasets while training in roughly one-sixth of the time and using roughly half the memory of the voting ensemble, indicating that a single boosted-tree model is a more favourable accuracy–cost operating point than the heavier alternatives for cloud-based intrusion detection.**

***Index Terms*—Intrusion Detection System, Cloud Computing Security, Machine Learning, Gradient Boosting, XGBoost, NSL-KDD, CICIDS2017, Lightweight Detection**

## I. Introduction

Cloud computing has become a foundational component of modern digital infrastructure, enabling organizations to store, process, and access large volumes of data at low marginal cost. The swift adoption of cloud platforms has helped industries improve scalability, elasticity and resource utilization. But this very adoption has also increased the attack surface . Cloud environments have become common targets for sophisticated intrusions and unauthorized access . Conventional methods of protection including firewalls and signature-based detection are getting less effective because they mostly recognize known attack signatures and are not generalizable to new or evolving threats. Thus, Intrusion Detection Systems (IDS) have become an integral part of cloud security by constantly monitoring network traffic for anomalous activity. Machine learning has improved the capabilities of IDS by allowing anomaly detection, pattern recognition and learning from large amounts of network data [1]–[6].

Despite this progress, there are a number of practical limitations. A significant fraction of recent work relies on deep neural architectures or metaheuristic optimization procedures that significantly increase computational cost and reduce reproducibility in realistic deployments [7]–[9].Other frameworks are tested on very large or proprietary data sets that are difficult to replicate, limiting cross study comparison. The community still needs lightweight and reproducible detection frameworks that achieve competitive accuracy without incurring heavy computational overhead [10]–[13].

This prompted the researcher, instead of increasing the complexity of the architecture, to investigate ways of reducing it. In this study we formulate a rather conservative question: *how much detection* The accuracy is really sacrificed with a single accurately programmed In this instance, do we use a gradient-boosted tree or a more weighty ensemble or hybrid model? Meta-path Classification (MPC) as a lightweight proposed labeler.Extreme Gradient Boosting (XGBoost) as a lightweight proposed detector and Meta-path Classification (MPC) as a


DOI: https://doi.org/10.67535/tsp.000003.019
Published under the ICACNC 2026 Proceedings

lightweight proposed labeler. Compare it, under the same preprocessing and evaluation methods, to one of the following: Random Forest baseline, a LightGBM model, along with a three-member soft-voting ensemble of them. Combination of Random Forest, Logistic Regression and Histogram Gradient Boosting. The following are the contributions of this work.

provide a controlled, reproducible comparison of four model families for cloud IDS across ten random seeds, reporting means, standard deviations and paired significance tests rather than single-run point estimates.

- We measure and report the full computational footprint—training time, inference time, memory, and CPU time—to support claims about lightweight deployment, an aspect frequently asserted but rarely quantified.
- We perform an ablation study isolating each ensemble component and a per-attack-class analysis, and we validate all findings on the modern CICIDS2017 dataset in addition to NSL-KDD.
- We demonstrate that the gradient-boosted tree is not only competitive but also achieves the best accuracy per cost, a metric which we propose is more useful than a marginal increment in accuracy for resource-constrained cloud nodes.

## II. Related Work

### A. Intrusion Detection in Cloud Computing

The distributed and multi-tenant nature of cloud environments creates security challenges that are poorly addressed by classical defenses. Masoodi in [1] studied machine-learning IDS in multi-cloud environments and achieved higher detection accuracy and fewer false alarms. Attou et al. [3] proposed a cloud IDS combining several machine-learning models and reported gains over conventional detection schemes. Al-Ghuwairi et al. [4] investigated time-series anomaly detection for cloud IDS. They found that learning-based models are well adapted to shifting threat patterns.

### B. Machine Learning Techniques for Intrusion Detection

Machine learning plays a key role in IDS because it can analyze high dimensional traffic statistics and surface latent patterns. Kaur and Kaur [2] trained a learning-based IDS for software defined cloud networks and reported faster attack identification with simplified network visibility. Samriya et al. [10] improved feature handling to enhance detection rates in cloud environments. Khan and Haroon [11] used ensemble learning with feature selection to enhance detection and minimize false alarms.

### C. Comparative Studies of ML Models for IDS

There have been some studies that directly compare families of algorithms. Ensemble and hybrid designs are observed to be superior to individual classifiers for cloud security as mentioned by Alhakeem and Ajlan [6]. Al Farsi et al. [14] emphasized the importance of data quality and model choice. Mahendar and Shivakanth [15] claimed that neural networks, SVM and Random Forest are the consistent among the models in cloud intrusion detection. These comparisons complement our work which explicitly relates accuracy to the measured computational cost.

TABLE I
Dataset Description

| Dataset | Samples | Features | Task |
|---|---|---|---|
| NSL-KDD (train) | 125,973 | 41 | binary |
| NSL-KDD (test) | 22,544 | 41 | binary |
| CICIDS2017 (sampled) | 99,899 | 78 | binary |

### D. Deep Learning and Hybrid IDS Approaches

Recent hybrid and deep approaches push accuracy at the expense of complexity. Mahdi Alhusseini et al. [7] combined artificial intelligence with metaheuristic optimization for cloud threat detection; Karahan et al. [8] used swarm-intelligence hyperparameter optimization with a deep model; and Garikapati et al. [9] proposed an explainable hybrid model. Ponnapalli et al. [16] introduced a hybrid learning model for cloud attacks. These methods are powerful but computationally demanding, which motivates our lightweight alternative.

### E. Feature Selection and Optimization Techniques

Feature selection and optimization are widely used to enhance IDS performance and reduce cost. Emirmahmutoğlu and Atay [13] introduced a framework of anomaly IDS driven by feature selection; Maazalahi and Hosseini [12] utilized metaheuristic optimization to feature selection for botnet detection; Olusesi et al. [17] integrated SVM with a fast firefly algorithm. While such optimization is effective, it comes with tuning overhead that our boosted-tree approach avoids with strong default regularization.

## III. Methodology

### A. Dataset Description

The main evaluation is conducted on the nsld-kdd benchmark widely used in the research of network intrusion detection [3], [11]. The NSL-KDD was selected because it removes the duplicate records of KDD'99, provides a balanced train/test difficulty, and is computationally manageable for controlled academic experimentation. Each record contains 41 features that describe connection and traffic behavior, and is labeled as either normal or attack. To evaluate generalization on a more modern threat landscape, we also evaluate on CICIDS2017 which contains modern attack types captured from realistic network traffic. The data splits are given in Table I.

### B. Data Pre-processing

Raw traffic records contain categorical and numerical attributes to be converted before training. The pipeline has three stages. We first label encode the categorical features (i.e., `protocol_type`, `service`, `flag`) to integer codes, with the encoders being fit on the combined train and test domains, to ensure a consistent code space. Second, feature–label separation offers a feature matrix $X$ and a binary target

(normal vs. attack). Third, infinite and missing values are removed from CICIDS2017 and it is downsampled to $10^5$ records, while preserving the class proportions, and then split into train and test partitions with a 80/20 ratio, maintaining the class balance between the partitions.

### C. Feature Scaling

Features are standardized to have zero mean and unit variance:

$$z = \frac{x - \mu}{\sigma}, \tag{1}$$

where $x$ is the original feature value, $\mu$ the mean value of the feature and $\sigma$ the standard deviation. Standardization stabilizes the linear and gradient-based parts of the compared models and improves convergence. Tree ensembles are scale-invariant but are trained on the same standardized inputs for a strictly identical protocol.

### D. Proposed Lightweight Model

The proposed detector is an Extreme Gradient Boosting (XGBoost) classifier. XGBoost builds an additive ensemble of regression trees in a stage-wise manner,

$$F_m(x) = F_{m-1}(x) + \gamma_m \, h_m(x), \tag{2}$$

where $h_m$ is the weak learner added at iteration $m$ with contribution $\gamma_m$. Each new tree is fit to the gradient of a regularized objective,

$$\mathcal{L} = \sum_i \ell(y_i, \hat{y}_i) + \sum_m \Omega(h_m), \qquad \Omega(h) = \tfrac{1}{2}\lambda \|w\|^2, \tag{3}$$

where $\ell$ is the The penalty function $\Omega$ gives an extra cost with respect to the leaf-weight magnitude $w$ and the logistic loss. Histogram-based split finding speeds up the training process, while at the same time using an external optimization loop is avoided due to the built-in $\ell_2$ regularization for strong generalization, and memory is minimal. These are intentionally standard and used in combination with 100 trees, a maximum depth of 6 and default learning rate; any advantage is then representative of the model family and not necessarily of being custom made for the particular dataset of interest.

### E. Baseline and Comparison Models

Three more families are assessed to provide a context for the proposed model. With identical preprocessing methods and seeds.

**Random Forest** is the single model baseline and is an ensemble The average of the decision trees summed up by majority voting:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_T(x)\}, \tag{4}$$

with $h_t$ the prediction of tree $t$ and $T$ the number of trees.

**LightGBM** is a second gradient-boosting implementation using leaf-wise growth, included to test whether the boosted-tree advantage is specific to one library.

**Soft-Voting Ensemble** combines Random Forest, Logistic Regression, and Histogram Gradient Boosting using equal weights. The ensemble averages class probabilities,

$$\hat{y} = \arg\max_c \sum_{i=1}^{N} w_i \, P_i(c \mid x), \tag{5}$$

where $P_i(c \mid x)$ is the probability from member $i$, $w_i$ its weight, and $N$ the number of members. We fix $w_i = 1$ for all members; this equal-weighting choice is stated explicitly because the weighting scheme is a frequent source of ambiguity in voting-based IDS work. This ensemble represents the heavier, more complex design against which the lightweight model is judged.

### F. Evaluation Protocol and Metrics

All models were trained and tested with 10 random seeds. ($\{42, 123, \ldots, 2021\}$); we report the mean and standard deviation of each metric. A paired Student's t-test is used to determine the statistical significance of differences in the accuracy. Per-seed scores averaged and compared by $t$-test. The performance is evaluated on the basis of accuracy and precision, recall, and F1-score:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}, \tag{6}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \qquad \text{F1} = \frac{2\,PR}{P + R}. \tag{7}$$

Precision, recall, and F1 are weighted by class support. To substantiate the lightweight claim, we additionally record wall-clock training time, inference time, peak memory increment, and CPU time for each model.

## IV. Results

### A. Experimental Setup

We performed experiments in Python using scikit-learn, XGBoost and LightGBM on a commodity workstation. All models are tested on the same encoded and standardized inputs, and the same ten-seed protocol, so that any difference can be assigned to the models, and not incidental preprocessing choices."

### B. Detection Performance

NSL-KDD results are presented in Table II. Among the detectors, the XGBoost detector has the highest accuracy (0.8009) and F1-score (0.7997), which is 2.6 accuracy points higher than the Random Forest baseline, and higher than both LightGBM and the soft-voting ensemble. Most importantly, the heavier voting ensemble (0.7945) does not outperform the single boosted tree, directly challenging the common assumption that ensemble complexity necessarily improves detection. The near-zero standard deviations for XGBoost and LightGBM are because of their deterministic behaviour under fixed data and seeds.

TABLE II
NSL-KDD PERFORMANCE OVER 10 SEEDS (MEAN ± STD)

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Random Forest | $0.7747_{\pm 0.0065}$ | 0.8358 | 0.7747 | 0.7718 |
| **XGBoost (Proposed)** | **0.8009** | **0.8488** | **0.8009** | **0.7997** |
| LightGBM | 0.7970 | 0.8468 | 0.7970 | 0.7956 |
| Voting Ensemble | $0.7945_{\pm 0.0030}$ | 0.8459 | 0.7945 | 0.7930 |

TABLE III
CICIDS2017 VALIDATION OVER 10 SEEDS (MEAN ± STD)

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Random Forest | $0.8400_{\pm 0.0008}$ | 0.8202 | 0.8400 | 0.8196 |
| **XGBoost (Proposed)** | **0.8499** | 0.8450 | **0.8499** | 0.8165 |
| LightGBM | 0.8479 | 0.8447 | 0.8479 | 0.8118 |
| Voting Ensemble | $0.8439_{\pm 0.0004}$ | 0.8517 | 0.8439 | 0.7996 |

The same ordering holds on the modern CICIDS2017 data (Table III), where XGBoost again achieves the best accuracy (0.8499). This consistency across a classic and a contemporary dataset indicates the finding is not an artifact of NSL-KDD. Figure 1 visualizes accuracy and F1 across both datasets.

### C. Statistical Significance

The paired $t$-test $p$-values for the differences in accuracy w.r.t. the voting ensemble are shown in Table IV. The difference is statistically significant on both data sets at $\alpha = 0.05$ showing that the difference between boosted tree and heavier ensemble is not noise, but systematic. This directly contravenes the belief that the absolute differences in the first few runs are not significant.

### D. Computational Footprint

The light-weight claim is substantiated in table V. The speed at which XGBoost trains is only 1.74 s (one sixth of the voting ensemble's 10.71 s) and the speed of its inference is the fastest among all models, and it has the least memory of all models (3.45 MB). The Random Forest baseline is more expensive and heavier than the boosted trees, and the voting ensemble is the costliest on all measures. Figure 2 illustrates the accuracy vs training time tradeoff; the operating point is clearly suggested since the proposed model is in the upper left hand corner (high accuracy, low cost) whilst the voting ensemble is stuck in the upper right hand corner (high cost, no accuracy gain).

### E. Ablation Study

Table VI isolates each member of the voting ensemble for a better understanding. Histogram Gradient Boosting alone (0.8081) outperforms not only the other single members but also the full voting ensemble (0.7945). In other words, a strong boosted tree, a weaker Random Forest and a linear model *dilute* performance rather than improve it. This motivates the preference for the proposed single boosted tree and reinforces the main message that further ensemble machinery is not justified here.

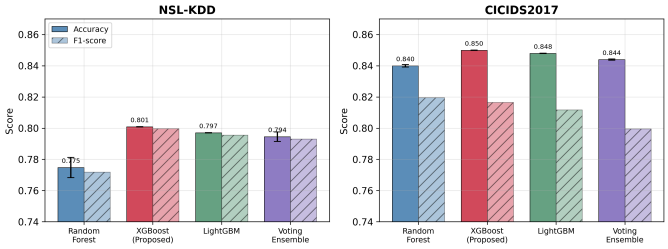


Fig. 1. Accuracy and F1-score on NSL-KDD (left) and CICIDS2017 (right). The proposed gradient-boosted tree leads on both datasets.

TABLE IV
PAIRED $t$-TEST $p$-VALUES (ACCURACY VS. VOTING ENSEMBLE)

| Model | NSL-KDD | CICIDS2017 |
|---|---|---|
| Random Forest | $2.70 \times 10^{-5}$ | $4.25 \times 10^{-7}$ |
| XGBoost (Proposed) | $1.27 \times 10^{-4}$ | $4.99 \times 10^{-12}$ |
| LightGBM | $3.66 \times 10^{-2}$ | $1.86 \times 10^{-10}$ |

### F. Per-Attack-Class Analysis

The per-class performance under the multiclass formulation is reported in Table VII and the normalized confusion matrix is depicted in Figure 4. The model is very good at detecting high-volume DoS traffic (F1 $=$ 0.87 ) and is also reliable at recognizing Normal traffic. However, the recall on the rare R2L and U2R classes is low. This is a well-known characteristic of NSL-KDD, where the test partition contains attack types and proportions not available in training. We report it openly because such minority classes are precisely where the future work on feature level is most valuable .

## V. DISCUSSION

The results of the experiments suggest that the best balance between accuracy and computation time for cloud-based intrusion detection on the benchmarks is obtained using a well-tuned gradient-boosted tree. It is the most accurate on both NSL-KDD and CICIDS2017, and the difference is statistically significant; It is not only the cheapest that we have to train it, but it is also one of the lightest in memory.

A noteworthy outcome is that ensemble complexity did not pay off. Both the three-member soft-voting ensemble and, on NSL-KDD, the Random Forest baseline are outperformed by a single boosted tree, and the ablation shows that the strongest ensemble member in isolation beats the ensemble as a whole. This aligns with the view that uninformative or weaker members can dilute a voting ensemble, and it cautions against equating architectural complexity with detection quality.Ourfinding corroborates comparative work that notes good tree-based Ourfinding's most notable results were the consistency of the findings and the reliability of the data. However, it is different from the performance of the random model [15], [18]. A greater focus on heavier hybrid/metaheuristic designs [7], [8]; there is a difference here, though: where we have computational resources. Cost being a secondary issue, not a first priority, just optimizing accuracy. [19], [20].

TABLE V
COMPUTATIONAL FOOTPRINT ON NSL-KDD

| Model | Train (s) | Infer (s) | Mem (MB) | CPU (s) |
|---|---|---|---|---|
| Random Forest | 5.51 | 0.104 | 7.60 | 19.50 |
| **XGBoost (Proposed)** | **1.74** | **0.028** | **3.45** | **5.29** |
| LightGBM | 1.53 | 0.055 | 4.97 | 4.72 |
| Voting Ensemble | 10.71 | 0.256 | 6.39 | 27.92 |

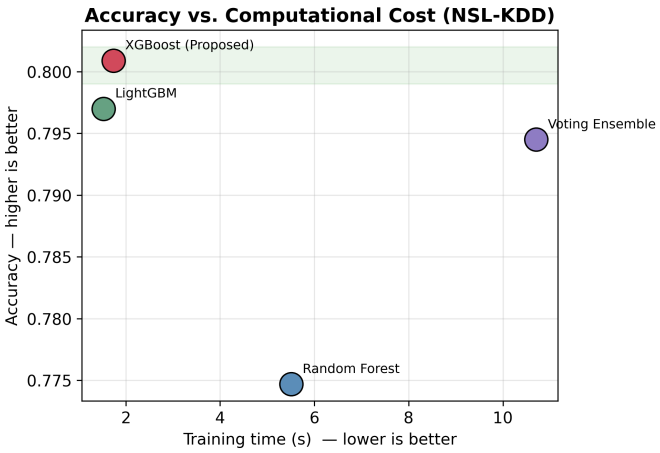


Fig. 2. Accuracy versus training time on NSL-KDD. The proposed model sits at the favourable upper-left operating point.

This has a direct practical implication for cloud deployment. Detection nodes in multi-tenant or edge-adjacent cloud settings are typically resource-constrained, and a model that trains in under two seconds and takes a few megabytes can be retrained often with traffic distribution drifts without the optimization overhead of metaheuristic or deep approaches [21], [22].

There are two implications for future work. The first is that on NSL-KDD, the distribution between the training and test data is not identical, which leads to a low recall on the rare classes: R2L and U2R. Targeted resampling or cost-sensitive learning is a logical and next logical step. Second, we test on two datasets and another test of generalization would be BoT-IoT or TON-IoT. The uniform results of the central conclusion on the accuracy/cost relationship do not suffer from either of these limitations.

## VI. CONCLUSION

We proposed a lightweight intrusion detection framework for cloud computing based on a single gradient-boosted tree, and extensively evaluated it against a random forest baseline model, a light gradient boosted machine (LightGBM) model and a soft voting ensemble of models with 10 random forest seeds and paired significance testing, full computational profiling, an ablation study and per-class evaluation on NSL-KDD and CICIDS2017. The boosted-tree detector outperformed the ensemble with statistically significant margins on both datasets, and trained approximately 6 times faster and used approximately half the memory. Based on the results, it can be concluded that, for In the cloud intrusion detection, the

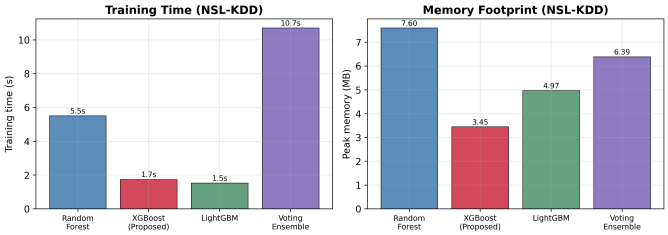


Fig. 3. Training time and peak memory by model on NSL-KDD.

TABLE VI
ABLATION OF ENSEMBLE COMPONENTS (NSL-KDD, 10 SEEDS)

| Component | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Random Forest | 0.7743 | 0.8358 | 0.7743 | 0.7714 |
| Logistic Regression | 0.7539 | 0.8063 | 0.7539 | 0.7516 |
| Histogram GB | **0.8081** | 0.8529 | 0.8081 | 0.8072 |
| Full Voting Ensemble | 0.7945 | 0.8459 | 0.7945 | 0.7930 |

cost constraint is the cloud resource, and a single well-regularized boosted is used. The more complex ensemble or hybrid is not as good an operating point as tree. designs. Future research will focus on recall of minority classes with a focus on costsensitivity. Validating learning and extending to other modern IoT and cloud datasets.

## REFERENCES


[1] H. J. K. A. Masoodi, "Evaluating the effectiveness of machine learning-based intrusion detection in multi-cloud environments," *Babylonian Journal of Internet of Things*, vol. 2024, pp. 94–105, 2024.

[2] G. Kaur and M. Kaur, "Software defined network-based intrusion detection in cloud environment using machine learning," *CAHIERS MAGELLANES-NS*, vol. 6, no. 2, pp. 7015–7029, 2024.

[3] H. Attou, A. Guezzaz, S. Benkirane, M. Azrour, and Y. Farhaoui, "Cloud-based intrusion detection approach using machine learning techniques," *Big Data Mining and Analytics*, vol. 6, no. 3, pp. 311–320, 2023.

[4] A.-R. Al-Ghuwairi, Y. Sharrab, D. Al-Fraihat, M. AlElaimat, A. Alsarhan, and A. Algarni, "Intrusion detection in cloud computing based on time series anomalies utilizing machine learning," *Journal of Cloud Computing*, vol. 12, no. 1, p. 127, 2023.

[5] B. R. Kikissagbe and M. Adda, "Machine learning-based intrusion detection methods in iot systems: A comprehensive review," *Electronics*, vol. 13, no. 18, p. 3601, 2024.

[6] M. S. Alhakeem and K. B. Ajlan, "A comparative evaluation of machine learning-based intrusion detection systems for securing cloud environments," *Journal of Information Security and Cybercrimes Research*, vol. 7, no. 2, pp. 127–142, 2024.

[7] M. Mahdi Alhusseini, A. Rouhi, and M.-R. Feizi-Derakhshi, "Ai-powered hybrid intrusion detection framework for cloud security using novel metaheuristic optimization," *arXiv e-prints*, pp. arXiv–2601, 2026.

[8] O. Karahan, B. Ataşlar-Ayyıldız, and P. Ayyıldız, "Network intrusion detection system using a hybrid deep learning model with swarm intelligence-based hyperparameter optimization," *The Journal of Supercomputing*, vol. 81, no. 15, p. 1346, 2025.

[9] H. Garikapati, K. Challapalli, S. V. Ramineni, V. M. S. Adusumilli, R. S. C. Kothamasu, and S. Anamalamudi, "An explainable ai-driven hybrid model for enhanced intrusion detection in network security," in *2025 Fifth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*. IEEE, 2025, pp. 1–7.

[10] J. K. Samriya, S. Kumar, M. Kumar, H. Wu, and S. S. Gill, "Machine learning-based network intrusion detection optimization for cloud computing environments," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 4, pp. 7449–7460, 2024.

TABLE VII
PER-ATTACK-CLASS PERFORMANCE (NSL-KDD, MULTICLASS)

| Class | Precision | Recall | F1 |
|---|---|---|---|
| Normal | 0.665 | 0.930 | 0.776 |
| DoS | 0.959 | 0.789 | 0.866 |
| Probe | 0.660 | 0.627 | 0.643 |
| R2L | 0.978 | 0.171 | 0.291 |
| U2R | 0.276 | 0.119 | 0.167 |

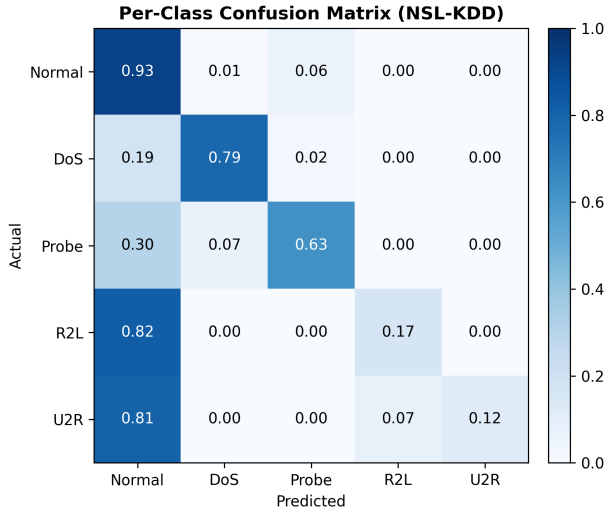


Fig. 4. Normalized per-class confusion matrix on NSL-KDD for the proposed model.


[11] M. Khan and M. Haroon, "Detecting network intrusion in cloud environment through ensemble learning and feature selection approach," *SN Computer Science*, vol. 5, no. 1, p. 84, 2023.

[12] M. Maazalahi and S. Hosseini, "Machine learning and metaheuristic optimization algorithms for feature selection and botnet attack detection," *Knowledge and Information Systems*, vol. 67, no. 4, pp. 3549–3597, 2025.

[13] E. Emirmahmutoğlu and Y. Atay, "A feature selection-driven machine learning framework for anomaly-based intrusion detection systems," *Peer-to-Peer Networking and Applications*, vol. 18, no. 3, p. 161, 2025.

[14] A. Al Farsi, A. Khan, M. M. Bait-Suwailam, and M. R. Mughal, "Comparative performance evaluation of machine learning algorithms for cyber intrusion detection," 2024.

[15] K. Mahendar and G. Shivakanth, "A performance analysis of ml-based intrusion detection systems in cloud environments," *International Journal of Electrical and Electronic Engineering & Telecommunications*, vol. 14, no. 4, pp. 243–252, 2025.

[16] S. Ponnapalli, R. R. Dornala, and K. T. Sai, "A hybrid learning model for detecting attacks in cloud computing," in *2024 3rd International Conference on Sentiment Analysis and Deep Learning (ICSADL)*. IEEE, 2024, pp. 318–324.

[17] A. T. Olusesi, I. K. Okakwu, A. A. Okubanjo, A. I. Oyedeji, and O. O. Bello, "Intrusion detection in cloud computing using support vector machine and fast firefly algorithm," *AUIQ Technical Engineering Science*, vol. 3, no. 1, p. 1, 2026.

[18] B. Song, "Random forest based intrusion detection system," in *2024 Asian Conference on Communication and Networks (ASIANComNet)*. IEEE, 2024, pp. 1–4.

[19] K. A. Maodah, S. Alhomdy, and F. Thabit, "Detecting intrusions in cloud-based ensembles: evaluating voting and stacking methods with machine learning classifiers," *Frontiers in Computer Science*, vol. 7, p. 1623375, 2025.

[20] F. Wang and S. Xie, "Cybersecurity in cloud computing ai-driven intrusion detection and mitigation strategies," *IEEE Access*, 2025.

[21] N. T. Nguyen and N. Nguyen, "Performance evaluation of neighbor-based learning models for network intrusion detection system," *Int. J. Management and Data Analytics*, vol. 5, no. 1, pp. 124–131, 2025.

[22] A. K. S. Ali, A. Raza, H. Arif, and A. A. Hussain, "Intelligent intrusion detection and data protection in information security using artificial intelligence and machine learning techniques," *Spectrum of Engineering Sciences*, pp. 818–828, 2025.