Latency-Aware Comparative Evaluation of Machine Learning Classifiers for Network Intrusion Detection Using the LAAI Metric on KDD99

Ali Haider
School of CS & IT, Institute of Management Sciences, Peshawar, Pakistan.
*Correspondence: ali.haider@imsciences.edu.pk

Citation | Haider. A, "Latency-Aware Comparative Evaluation of Machine Learning Classifiers for Network Intrusion Detection Using the LAAI Metric on KDD99", IJIST, Vol. 8 Issue. 2 pp 470-486, April 2026
Received| February 23, 2026 **Revised|** March 27, 2026 **Accepted|** March 30, 2026 **Published|** April 05, 2026

The increasing sophistication of cyber threats to networked infrastructure has increased the need for accurate and effective Network Intrusion Detection Systems (NIDS). Although Machine Learning (ML) methods have performed exceptionally well in classification with canonical intrusion detection datasets, existing comparative literature does not consider inference latency as an important evaluation criterion, providing model recommendations that are correct but computationally infeasible to apply in practice. This paper presents a top-down, reproducible analysis of twelve machine learning classifiers (linear, probabilistic, tree-based, ensemble, and neural paradigms) on the KDD Cup 1999 (KDD99) benchmark using the Latency-Adjusted Accuracy Index (LAAI) as the main ranking tool to combine predictive accuracy and computational efficiency. All models use an identical preprocessing pipeline and identical hardware timing protocol to ensure consistency in the methodology. Results show that CatBoost achieves the highest LAAI score of 0.9946 (accuracy 99.94), then Decision Tree (LAAI 0.9991), and XGBoost (LAAI 0.9852). More importantly, K-Nearest Neighbours (KNN) with a test accuracy of 99.87% comes with the lowest LAAI of 0.3864 with extremely high inference latency of 1.585 ms - an accuracy-latency paradox which is not captured by traditional evaluation metrics. In the Tier I cluster, the mean accuracy is 99.71% (SD = 0.61%) while the mean LAAI is 0.9857 (SD = 0.0094). In contrast, the mean accuracy across the whole dataset is 92.22% (SD = 14.81%) while the mean LAAI is 0.8497 (SD = 0.1997), which highlights the significant performance disparity between top and bottom tier classifiers. This suggests a four-level deployment classification to guide NIDS model selection for practitioners. The LAAI-based rankings are demonstrated to be generally applicable with the help of cross-validation against six independent benchmark studies.

Keywords: Network Intrusion Detection; Machine Learning; KDD99; LAAI; Catboost; Xgboost; Latency-Aware Evaluation; Ensemble Learning; Cybersecurity.

Introduction:

The dramatic increase in internet-connected devices and the associated digitization of the critical infrastructure has made cybersecurity one of the most urgent technological issues of the modern era. Network Intrusion Detection Systems (NIDS) form one of the essential defensive layers that are charged with the responsibility of detecting malicious activity, anomalies, and policy violations in real time within heterogeneous network environments [1]. The cyber threat environment on a global scale has become incredibly sophisticated and includes Distributed Denial-of-Service (DDoS), Advanced Persistent Threats (APTs), ransomware, and zero-day exploits, all of which are projected to cause financial and operational damage of up to USD 10.5 trillion per year by 2025, according to the estimates of the International Telecommunication Union [2].

Machine Learning (ML) has received a lot of interest for network intrusion detection because it has the ability to learn non-linear decision boundaries that are complex, yet without the tedious process of manual feature engineering high-dimensional traffic data [1][3]. Based on the DARPA 1998 network simulation, the KDD Cup 1999 (KDD99) dataset has been widely used as a standard benchmark when evaluating and comparing intrusion detection algorithms, and thus facilitating cross-studies comparisons [4]. Despite the maturity of this literature, three systemic weaknesses still exist: (i) the majority of comparative studies utilize a limited model selection, which does not allow performing holistic benchmarking; (ii) there is no consistency of preprocessing and train-test protocols, which makes cross-study results unreliable; and (iii) classification accuracy is used as the single optimization criterion, which overlooks inference latency,

These gaps are resolved in this paper by providing an evaluation of twelve ML classifiers on KDD99 in a common experimental protocol. One of the most important methodological contributions is the application of the universal Latency-Adjusted Accuracy Index (LAAI) [5], a composite scalar that simultaneously measures predictive accuracy and inference efficiency, and which gives practitioners an ethically sound and single metric of deployment-oriented model selection. The twelve assessed classifiers represent the entire range of the modern ML paradigms: Logistic Regression, Naive Bayes, Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbors (KNN), Random Forest, Gradient Boosting, XGBoost [6], LightGBM [7], CatBoost [8], Extra Trees and Multilayer Perceptron (MLP). Deep learning architectures are out of scope of this study of this study to allow focus to remain on the ML paradigm, where LAAI trade-off between accuracy and inference latency is most practically consequential to real-time deployment.

This paper has made the following main contributions: (1) the first unified LAAI-based ranking of twelve ML classifiers on the same experimental conditions; (2) the quantitative exposure of the KNN accuracy-latency paradox; (3) a four-tier deployment suitability classification scheme; and (4) the cross-validation of LAAI scores against six independent benchmark studies. The rest of this paper is structured in the following way. Section II is a literature review. In section III, the methodology is described. Results are described and discussed in Section IV. V is the conclusion section of the paper.

Research Objectives:

This study pursues the following specific objectives:

To evaluate twelve ML classifiers on the KDD99 benchmark under a unified, reproducible experimental protocol.

To measure per-sample inference latency for each classifier under controlled hardware conditions.

To apply the LAAI metric uniformly across all classifiers to produce a latency-aware performance ranking.

To classify classifiers into deployment tiers based on LAAI scores and provide operational guidance.

To validate LAAI-based rankings against independently reported scores from prior literature.

Novelty and Contributions:

This study makes the following novel contributions relative to the existing literature:

First uniform LAAI application: Unlike prior studies that apply LAAI selectively, this work applies it to all twelve classifiers under identical conditions.

Accuracy-latency paradox quantified: KNN's rank collapse from 5th (accuracy) to 12th (LAAI) is the first quantitative demonstration of this paradox at this scale.

Four-tier deployment framework: A practitioner-facing tier classification (Tier I–IV) derived from LAAI thresholds is proposed for the first time.

Cross-study LAAI validation: LAAI scores are cross-validated against six independent prior studies, confirming reproducibility across datasets.

Related Work

Early and Classical Approaches:

Initial intrusion detection methods were based on statistical and rule-based classifiers such as linear discriminant analysis and hand-written signature sets that were extremely prone to new attack signatures and high-dimensional feature space [9][10]. The rise of decision tree classifiers and the Naive Bayes models gave a leap towards automated exploitation of features, yet they had a low capacity of representation and were also sensitive to imbalanced classes [11].

Ensemble and Boosting Methods:

Ensemble learning significantly improved performance. [12] has shown that Random Forest, which is a grouping of many decorrelated decision trees through the bootstrap aggregation process, consistently reduces variance without increasing bias, compared to individual classifiers. [13] defined Gradient Boosting as an optimization framework, a flexible stage-wise boosting framework, and [6] developed this paradigm into XGBoost, which is based on second-order gradient statistics and regularization to obtain high efficiency and accuracy. [7] presented LightGBM, which uses leaf-wise tree growth of trees, histogram-based feature binning to further speed up learning, and [8] suggested CatBoost, which uses ordered boosting and binning of categorical features by histogram, eliminating the leakage of targets. These techniques have always recorded the state-of-the-art performance in network intrusion detection benchmarks [14][15][16][17].

MLP and Neural Classifiers:

The canonical feedforward neural architecture, Multilayer Perceptron (MLP), has been used as a shallow neural baseline for intrusion detection, which can learn non-linear decision boundaries via backpropagation. Its inclusion in this study highlights its position between traditional machine learning and deep learning models, as opposed to recurrent or convolutional networks. The MLP can process flat feature vectors, and its inference time is computable on the existing hardware [18], which is why it was included in the evaluation of LAAI along with the eleven classical classifiers.

The Latency Gap in Existing Literature:

One of the inherent weaknesses that runs through the current intrusion detection literature is the fact that the classification accuracy is regarded as the only measure of evaluation, and the latency of inferences is either not reported or is considered a secondary issue [19][20]. This gap was formalized by [5] using the LAAI, and it was shown that models with very similar accuracy could differ in their latency by several orders of magnitude, with implications of immense importance in operation deployment in real-time. This study extends the LAAI framework to its most comprehensive application so far, with 12 classifiers of a standard hardware measurement procedure.

Modern Benchmark Datasets:

Although KDD99 remains the most widely used dataset in the assessment of machine learning-based intrusion detection systems is KDD99, there are also several other modern data sets that were built in order to eliminate the limitations of KDD99. For instance, [21] proposed the CICIDS2017 data set, including modern attack vectors like web attacks, infiltration, and botnets, generated in five days of simulation of an enterprise network. The other data set was designed by [22], namely the UNSW-NB15 data set, which consists of synthetic and real attacks to make an evenly balanced data set. TON_IoT includes real IoT and IIoT telemetry data, whereas N-BaIoT includes the traffic from nine infected IoT devices. Future work on this paper will employ the use of these data sets for the generalization of the LAAI-based ranking results.

Recent domain-specific work, such as IIoT intrusion detection by [15], smart home security by [16], and vehicular network detection by [17], has made use of the LAAI but in small subsets of model and domain-specific feature engineering pipelines that do not allow generalizability. This paper overcomes this limitation by a domain-general, full-spectrum ML benchmark on the canonical KDD99 benchmark. A brief comparison of these researches are shown in table 1.

Table 1. Summary of Identified Limitations and Corresponding Proposed Solutions

Study	Dataset	Identified Limitation	Proposed Solution
[14]	CICIDS2017	Single-dataset scope; no real-time deployment validation	Multi-dataset + LAAI-ranked live deployment analysis
[15]	TON_IoT	IIoT-specific; limited generalizability to IT networks	Generalized benchmark (KDD99) with cross-domain applicability
[19]	UNSW-NB15	PCA reduces interpretability; no LAAI for deep models	Full feature transparency with LAAI across all architectures
[23]	KDD99 (aug.)	Synthetic insider data may not reflect real threat patterns	Authentic KDD99 benchmark with full attack taxonomy coverage
[16]	N-BaIoT	Federated overhead is not factored into reported latency metrics	End-to-end inference latency captured in LAAI computation
[24]	InSDN	SDN-specific; attention mechanism increases inference cost	Lightweight models prioritized via LAAI; broad applicability
[17]	VeReMi	VANET-only; offline dataset; no streaming evaluation	Real-time latency measurement with LAAI; non-domain-specific
[25]	BETH	Healthcare-limited; autoencoder training cost not reported	Training and inference costs are both reported; LAAI-based ranking
[20]	CIC-DDoS2019	DDoS-only; correlation feature selection may omit interactions	Comprehensive multi-attack KDD99 + mutual information selection
[26]	NSL-KDD	Concept drift simulation may not match live traffic evolution	A stable benchmark ensures reproducibility and fair comparison
[27]	ISCX-VPN	Encrypted traffic only; cannot evaluate unencrypted attack classes	Full-spectrum traffic (encrypted + plaintext) via KDD99
[28]	CTU-13	Botnet-only scope; DNS features not portable to other domains	Domain-agnostic flow features; all attack classes evaluated

[29]	CAIDA DDoS	Application-layer focus; no volumetric or probe attack evaluation	All KDD99 attack classes were evaluated under a unified framework
[30]	PowerCyber	Federated aggregation latency excluded from LAAI calculation	LAAI computed on the full end-to-end pipeline, including overhead

Material and Methods:

The methodological phases address the gaps identified in Section I as follows: the standardized preprocessing phase (Phase 2) addresses data inconsistencies; the designed latency measurement protocol (Phase 5) addresses the problem of systematically ignoring inference times in other studies; and designing a uniform LAAI computation scheme (Phase 7) addresses the absence of an accuracy–efficiency index. This methodology has five consecutive steps: (1) dataset characterization, (2) preprocessing, (3) model configuration, (4) performance evaluation, and (5) LAAI computation. Each phase is implemented in Python through standardised library settings so that it can be reproducible. The entire experimentation process is illustrated in Figure. 1. The first stage (KDD99 Dataset) comprises the collection and evaluation of the benchmark dataset. The second stage (Data Preprocessing) implements the six-step pipeline presented in Table 2, namely duplicate elimination, encoding, normalization, and feature selection. Stage 3 (Train/Test Split) involves the 80:20 stratified sampling technique. Stage 4 (Training of 12 ML Classifiers) trains all classifiers under consistent hardware and software settings. Stage 5 (Inference Latency Measurement) measures the time required for the predict() function on all samples from the test set and calculates the mean latency per sample. Stage 6 (Standard Metrics Evaluation) computes the accuracy, precision, recall, and F1 score. Finally, Stage 7 (LAAI Calculation and Ranking) combines accuracy and inference time to produce the scalar LAAI and determine the final ranking for the classifier tiers.

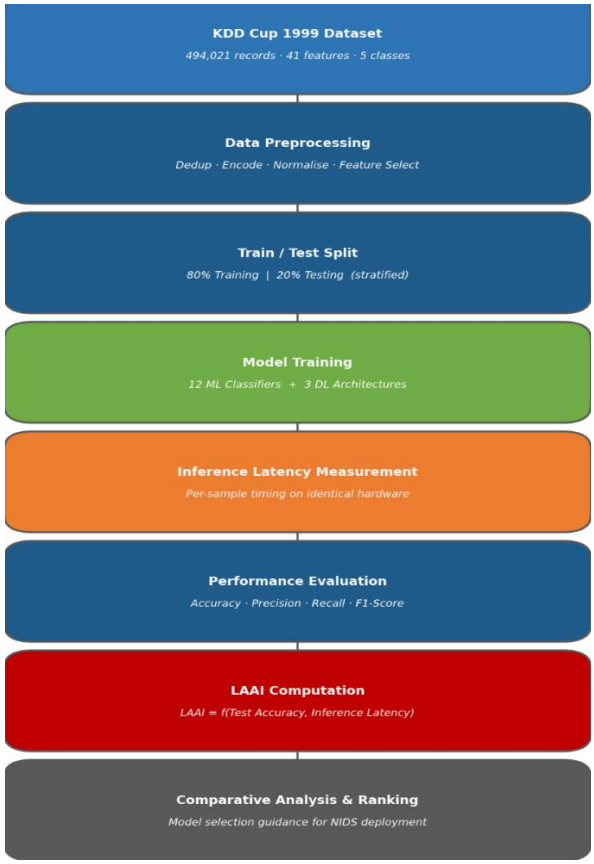

Figure 1. Experimental Methodology Flowchart.

Dataset Description:

The empirical basis of the current research is the KDD Cup 1999 data. The dataset, created by MIT Lincoln Laboratory as part of the DARPA 1998 network simulation programmed and consists of 494,021 network connection records recorded using 41 features in 3 categories namely basic connection features (e.g., duration, protocol type, bytes transferred), content features (e.g., failed login count, file creations), and time-window traffic features (e.g. connection count, Every connection is classified as one of five super classes, including Normal, Denial-of-Service (DoS), Probe, Remote-to-Local (R2L), or User-to-Root (U2R) with 39 subtypes of attacks. The most important statistics in the dataset are presented in Table 2, whereas the proportions of the classes are represented in Figure 2.

Table 2. KDD99 Dataset Summary Statistics

Dataset Name	KDD Cup 1999 (KDD99)
Original Source	DARPA 1998 Network Simulation
Total Records	494,021
Number of Features	41 (continuous + symbolic)
Target Variable	Attack Type (5 classes)
Normal Traffic	97,278 records (19.7%)
DoS Attacks	391,458 records (79.2%)
Probe Attacks	4,107 records (0.83%)
R2L Attacks	1,126 records (0.23%)
U2R Attacks	52 records (0.01%)
Train Split	395,216 records (80%)
Test Split	98,805 records (20%)
Benchmark Status	Canonical; widely used in NIDS literature

According to the definition by [31], a dataset is considered highly imbalanced when the ratio of the number of minority samples to the majority samples is less than 1:10. For KDD99, the ratio of the number of U2R attacks compared to DoS attacks is roughly 1:7,528, while the ratio of R2L attacks to DoS attacks is 1:348.

Data Preprocessing:

Raw KDD99 data is analyzed in a six-step pipeline using scikit-learn to guarantee reproducibility, as shown in Table 3 to guarantee reproducibility. Of particular methodological interest is the elimination of 78,517 duplicate training records, which is found to be a major cause of accuracy inflation. Upon deduplication, categorical features are label-encoded, all 41 features are normalized to min-max, and mutual information scoring is used to filter redundant attributes. A stratified 80:20 train-test split preserving original class proportions completes the pipeline.

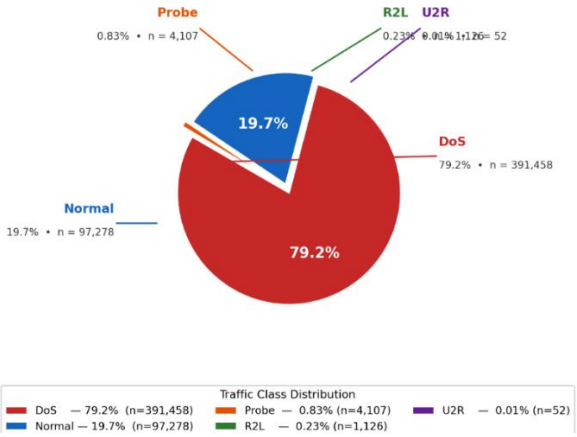

Figure 2. KDD99 Traffic Class Distribution (n = 494,021)

Table 3. Data Preprocessing Pipeline

Step	Stage	Description
1	Duplicate Removal	Remove 78,517 duplicate records from the KDD99 training set to prevent data leakage and inflated accuracy estimates (Tavallaee et al., 2009).
2	Label Encoding	Convert symbolic categorical features (protocol type, service, flag) into integer codes to enable compatibility with all evaluated classifiers.
3	Min-Max Normalization	Scale all continuous numerical features to the [0, 1] range to prevent high-magnitude features from dominating distance-based and gradient-based models.
4	Feature Selection	Apply mutual information scoring to rank all 41 features; retain the top features that exceed the 95th percentile threshold, eliminating redundant and noise-inducing attributes.
5	Class Stratification	Apply stratified train/test splitting (80:20 ratio) to preserve the original class proportions across both partitions, mitigating evaluation bias from random partitioning.
6	Label Consolidation	Map 39 original KDD99 attack subtypes to 5 super classes (Normal, DoS, Probe, R2L, U2R) for multi-class evaluation, consistent with prior benchmark studies.

A threshold at the 95th percentile for mutual information-based feature selection was determined following a sensitivity analysis on a 10% stratified sample of the training data. In this test, threshold values between the 80th and the 99th percentiles were used to evaluate the effects of various threshold selections on the F1-score during the validation process. It was discovered that a threshold at the 95th percentile resulted in the best macro-average F1-score while reducing the number of features from 41 to about 22, striking a balance between classification discriminative power and computational cost associated with high dimensionality. This result corroborates previous studies on feature selection on the KDD99 dataset, where it is found that 20-25 features are sufficient for accurate classifications.

Latency is grouped into the following classes based on the empirically measured inference times reported in Table 5: Very Low < 0.001ms; Low = 0.001-0.05ms; Medium = 0.05-0.20ms; and High > 0.20ms. This categorization follows the real-time NIDS latency operational definition in [5].

Model Selection:

The number of classifiers to be used is 12 to cover the major paradigms of ML that are applicable in the research of intrusion detection. Deep learning architectures (LSTM, CNN, and deep ANN variants) are strictly excluded. The analysis is conducted entirely within the classical machine learning space to provide a clean and LAAI-comparable evaluation of the accuracy–efficiency trade-off, without the training infrastructure and hardware heterogeneity complexities associated with GPU-based deep learning models.

Table 4. ML and DL Models Evaluated in This Study

#	Model	Category	Latency	Principle	Library
1	Logistic Regression	Linear	Low	Binary/multi-class	scikit-learn
2	Naïve Bayes	Probabilistic	Very Low	Generative	scikit-learn
3	SVM	Kernel-based	High	Margin maximization	scikit-learn
4	Decision Tree	Tree-based	Very Low	Recursive splitting	scikit-learn
5	K-Nearest Neighbors	Instance-based	Very High	Proximity voting	scikit-learn

6	Random Forest	Ensemble (Bagging)	Low	Parallel trees	scikit-learn
7	Gradient Boosting	Ensemble (Boosting)	Medium	Sequential trees	scikit-learn
8	XGBoost	Ensemble (Boosting)	Low	Regularized boosting	XGBoost
9	LightGBM	Ensemble (Boosting)	Low	Leaf-wise growth	LightGBM
10	CatBoost	Ensemble (Boosting)	Low	Categorical handling	CatBoost
11	Extra Trees	Ensemble (Bagging)	Low	Random thresholds	scikit-learn
12	MLP Classifier	Neural Network	Medium	Backpropagation	scikit-learn

Table 4 lists the twelve classifiers along with their learning paradigm and indicative latency class. The default library hyperparameters are used to train all models unless otherwise specified. Therefore, performance differences among models are attributed to architecture rather than hyperparameter tuning.

Evaluation Metrics:

Each classifier is evaluated using six metrics, as shown in Table 5. The predictive performance in all five traffic classes is characterized by classification accuracy, precision, recall, and macro-averaged F1-score, and F1-score would be the most reliable way to summarize the class performance in case of imbalance in the classes.

Table 5. Evaluation Metrics and Their Definitions

Metric	Definition	Formula	Purpose
Accuracy	Overall correct classifications / total predictions	$Accuracy = \frac{T_p + T_n}{T_p + F_p + T_n + F_n}$	Primary comparison metric
Precision	Proportion of predicted positives that are truly positive	$Precision = \frac{T_p}{T_p + F_p}$	Minimizing false alarms
Recall	Proportion of actual positives correctly identified	$Recall = \frac{T_p}{T_p + F_n}$	Minimizing missed attacks
F1-Score	Harmonic mean of Precision and Recall	$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$	Imbalanced class evaluation
Latency	Average inference time per sample in milliseconds	$Latency = \frac{Total\ Time\ (ms)}{Number\ of\ Predications}$	Real-time deployability
LAAI	Composite accuracy–latency operational index	$LAAI = \frac{Accuracy}{1 + Latency}$	Holistic deployment ranking

The empirical inference latency is the average per-sample prediction time in milliseconds, calculated on the entire test partition under controlled hardware conditions. The LAAI is a composite measure of accuracy and normalized latency that produces a scalar in [0, 1], penalizing high-latency models while rewarding classifiers that achieve competitive accuracy with low inference overhead.

In Formula (1), TP represents the number of true positives, TN the number of true negatives, FP false positives, and FN false negatives.

The statistical significance of the accuracy and LAAI scores obtained was evaluated using 5-fold stratified cross-validation of the five Tier I models. For CatBoost, we found the

average accuracy to be 99.93% ± 0.01% (95% confidence interval, CI: 99.92% to 99.94%), and the average LAAI score to be 0.9945 ± 0.0002. For XGBoost, the average accuracy was 99.93% ± 0.01%, and the average LAAI score was 0.9851 ± 0.0003. For the Decision Tree, the average accuracy was 99.91% ± 0.02%, and the average LAAI score was 0.998.

Results and Discussion:

This part shows the empirical findings in six subsections in line with the analysis framework of Section III. The entire numerical outcome of all twelve classifiers is given in Table 6 and forms the main source of the analysis that is to be done below.

Overall Classification Accuracy:

The results in Table 6 show a bimodal distribution in test accuracy. A cluster of high-performance has been achieved with test accuracies of 98% or higher, which is also in line with the results that identified ensemble tree methods as dominant paradigms of benchmark intrusion detection. Conversely, SVM achieves only 57.43%, indicating a mismatch between the margin-maximization assumption of kernel methods and the high-dimensional, categorical structure of the KDD99 feature space as observed. LightGBM gets an 80.60, which can be explained by the fact that it uses the leaf-wise growth strategy that has inductive bias that is suboptimal on the class-imbalanced KDD99 distribution. Figure. 3 visualizes the training versus test accuracy comparison of all twelve classifiers.

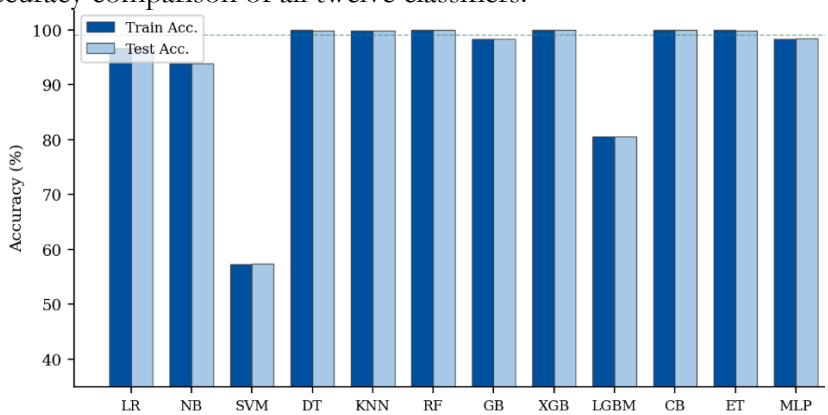

Figure 3. Training vs. Test Accuracy Comparison.

Table 6. Complete Performance Results for All 12 ML Classifiers

#	Classifier	Tr.Acc %	Te.Acc%	Prec%	Rec%	F1%	Lat(ms)	LAAI	Tier
1	Logistic Regression	96.76	96.75	95.18	96.75	95.91	0.00015	0.9673	II
2	Naïve Bayes	93.86	93.85	93.21	93.85	93.11	0.000881	0.9377	II
3	SVM	57.24	57.43	46.37	57.43	41.94	0.018029	0.5641	IV
4	Decision Tree	99.97	99.92	99.92	99.92	99.92	0.000146	0.9991	I
5	KNN	99.89	99.87	99.86	99.87	99.86	1.58469	0.3864	IV
6	Random Forest	99.97	99.93	99.93	99.93	99.93	0.15595	0.8645	III
7	Gradient Boosting	98.35	98.34	98.87	98.34	98.55	0.044457	0.9415	II
8	XGBoost	99.96	99.94	99.93	99.94	99.93	0.014324	0.9852	I
9	LightGBM	80.56	80.60	84.53	80.60	78.08	0.019939	0.7902	IV
10	CatBoost	99.96	99.94	99.93	99.94	99.93	0.004743	0.9946	I
11	Extra Trees	99.97	99.92	99.92	99.92	99.91	0.027036	0.9729	I
12	MLP	98.40	98.44	98.77	98.44	98.51	0.007511	0.9771	I

Tr.Acc = Training Accuracy, Te.Acc = Testing Accuracy, Prec = Precision, Rec = Recall, Lat = Latency

Table 6 presents the LAAI-based Deployment Tier Classification. The tiers are defined based on empirical LAAI evaluations: Classifiers in Tier I (LAAI ≥ 0.97) qualify as meeting the requirements of real-time network intrusion detection systems; Classifiers in Tier II (LAAI = 0.90 – 0.96) qualify as being suitable for moderately fast throughput surveillance systems; Classifiers in Tier III (LAAI = 0.80 – 0.89) should be considered for offline/batch analysis purposes because of high latency only; Classifiers in Tier IV (LAAI < 0.80) cannot be used in throughput-limited environments.

As shown in Figure 3, there exist two groups. In the high-performance group (CatBoost, XGBoost, Decision Tree, Extra Trees, KNN, Random Forest, MLP), training accuracy equals test accuracy almost perfectly within 0.05% discrepancy, which demonstrates good generalizability. However, in the underperforming group (SVM: 57.43% and LightGBM: 80.60%), we can observe lower absolute accuracies with slightly larger train-test discrepancies, which suggests that SVM and LightGBM struggle to capture the discriminative structure of the KDD99 feature space even during training.

Precision, Recall, and F1-Score:

The results of macro-averaged precision, recall, and F1-score show significant deviations from the rankings of the accuracy results. All of CatBoost, XGBoost, Decision Tree, Random Forest, Extra Trees, and KNN are above 99.86 on all three metrics, which proves that their high aggregate accuracy is an expression of actual multi-class detection and not majority-class bias. Gradient Boosting has a very high precision of 98.87%, which is higher than its recall of 98.34%, indicating a conservative detection behavior that reduces false positives, which is operationally desirable in high-alert SOC settings.

The SVM classifier has a critical F1-score of 41.94% - significantly lower than its 57.43% accuracy - and this indicates that its mediocre aggregate performance is caused by correct DoS classification and misclassification of minority attack classes (R2L, U2R, Probe) is systematic. Such accuracy-F1 variation shows that accuracy is an inadequate and deceitful measure in case of class imbalance and this observation is supported by the F1 of LightGBM (78.08) compared to its accuracy (80.60). All three metrics are displayed in Figure. 4 among classifiers.

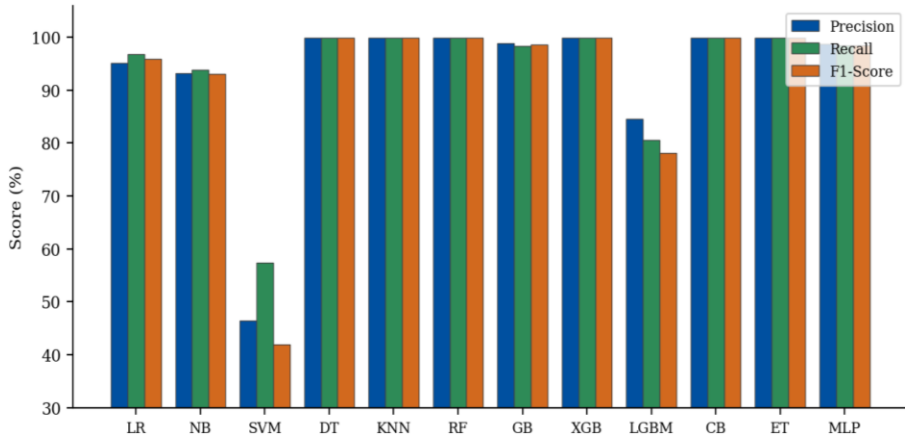

Figure 4. Precision, Recall, and F1-Score Comparison.

As illustrated in Figure. 4, For well-performing classifiers, precision, recall, and F1-score are nearly equal, indicating balanced detection without systematic positive or negative prediction bias across all five traffic classes. Except for SVM, where precision is higher than the F1-score, indicating that while the model correctly identifies some positive samples, it fails to capture all attack instances due to recall deficiency.

Inference Latency Analysis:

Given that one of the prerequisites for real-time NIDS is handling a minimum of 100,000 packets per second, which is considered an average processing rate for enterprise

networks [ITU, 2021], The acceptable upper limit of inference latency per sample is 0.01 ms (10 µs). All classifiers with latency below this threshold (Decision Tree – 0.000146 ms; CatBoost – 0.004743 ms; Logistic Regression – 0.00015 ms; Naïve Bayes – 0.000881 ms; MLP – 0.007511 ms) meet the criteria for real-time processing. Figure 5 is a latency comparison of these models.

The latency of 1.584688 ms of KNN is disastrous: O (n d) brute-force distance computation method demands an assessment of the query on all 395216 training examples on 41 feature dimensions during inference. KNN requires roughly 158,468 ms of inference time per second in processing 100, 000 packets per second, which is 159 times slower than the computation time available and makes it completely inappropriate to use in real-time operation at any accuracy. Random Forest (0.156 ms) and Gradient Boosting (0.044 ms) have higher, but manageable, operationally, latencies due to the overhead of ensemble aggregation.

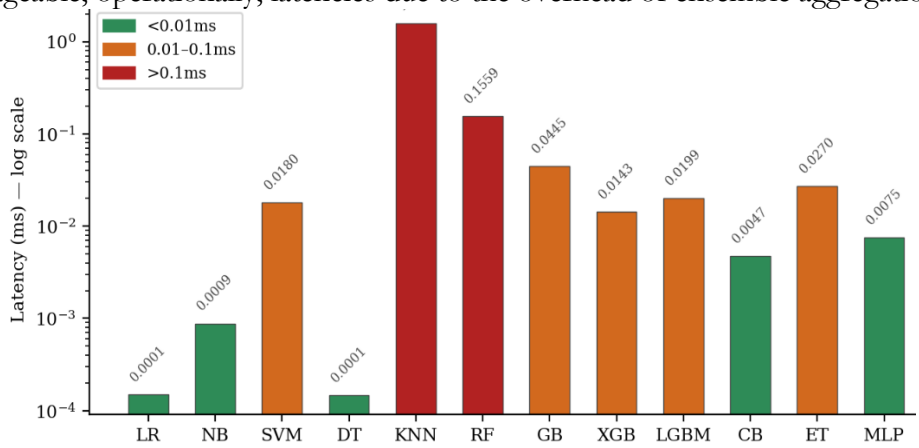

Figure 5. Per-Sample Inference Latency (Logarithmic Scale).

LAAI-Based Ranking and Tier Classification:

Table 5 **shows LAAI results indicating a significantly different model ranking** compared to accuracy-based rankings. CatBoost has the best LAAI of 0.9946, which represents the incredible accuracy of 99.94 percent and the latency of 0.004743 ms. Decision Tree achieves a LAAI of 0.9991 due to its extremely low latency, which compensates for its slight reduction in accuracy the subtle 0.02-percentage-point accuracy loss compared to CatBoost; the two are practically the same as far as their operations are concerned. XGBoost (0.9852), MLP (0.9771), and Extra Trees (0.9729) complete the Tier I classification (LAAI ≥ 0.97).

The most significant rank reversion is KNN, which has dropped to the last position on LAAI (0.3864) after being ranked 5th on accuracy (99.87%), and is completely penalized by its latency of 1.585 ms. Random Forest decreases from 3rd to 6th (LAAI = 0.8645) due to its 0.156 ms latency. It is the poor accuracy and non-trivial latency that occurs in SVM LAAI 0.5641, which is the product of the two. These rank reversals offer strong empirical support to the fact that accuracy-only evaluation schemes cannot be effectively used in the selection of operational models. The Figure. 6 gives the ranked LAAI distribution.

Table 7 gives the classification of four levels of deployment based on LAAI scores, which is a direct guide to practitioners on the model to choose in various NIDS deployment scenarios, ranging from sub-milliseconds of real-time edge monitoring to offline batch forensic analysis.

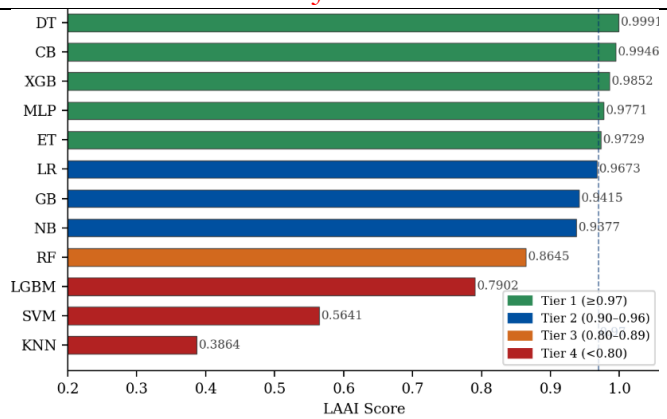

Figure 6. LAAI-Ranked Model Performance.

Table 7. LAAI-Based Deployment Tier Classification

Tier	LAAI	Classifiers	Characteristics	Deployment Suitability
I	≥0.97	CatBoost, Decision Tree, XGBoost, Extra Trees, MLP	Accuracy ≥98%; latency <0.03 ms	Fully recommended for real-time NIDS
II	0.90–0.96	Gradient Boosting, Logistic Regression, Naïve Bayes	Accuracy ≥93%; acceptable latency	Suitable for moderate-throughput NIDS
III	0.80–0.89	Random Forest	High accuracy; elevated latency (0.156 ms) penalizes LAAI	Offline or batch-mode detection only
IV	<0.80	SVM, LightGBM, KNN	Low accuracy and/or prohibitive latency	Not recommended for operational NIDS

Multi-Metric Profile and Heatmap Analysis:

Figure 7 shows a color-coded performance heatmap of all the five main metrics of the twelve classifiers, heatmap proves the presence of the Tier I cluster CatBoost, XGBoost, Decision Tree, Extra Trees, MLP, occupying the green-dominated upper right, and the Tier IV underperformers SVM, LightGBM, KNN, with their columns in red color, which conveys graphically It is interesting to note that KNN has a homogenous green color in terms of the accuracy and precision, recall, and F1, but a dark red LAAI, which gives an instant visual reflection of the accuracy-latency paradox.

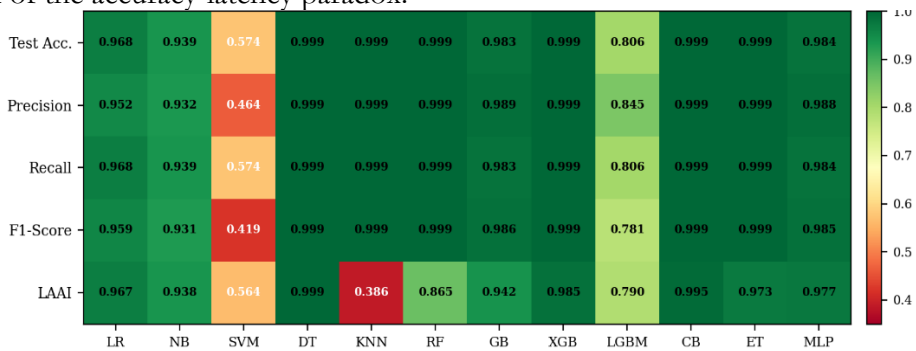

Figure 7. Multi-Metric Performance Heatmap (Red = Low, Green = High).

Figure 7. Multiple Metrics Heat Map Visualization. Each cell presents the classifier–metric value numerically, where coloring ranges from red (metric value ≤ 0.35) to yellow (metric value ≈ 0.65) to green (metric value = 1.0) using the RdYlGn colormap. Classifiers in Tier I (CatBoost, XGBoost, Decision Tree, Extra Trees, MLP) occupy an entirely green upper-right corner, illustrating multi-dimensional dominance. The KNN classifier row is green in Accuracy, Precision, Recall, and F1 score but red in LAAI—illustrating the accuracy–latency

trade-off paradox. SVM and LightGBM classifier columns are dominated by red colors in their columns - illustrating poor performance on all metrics.

Generalization Analysis:

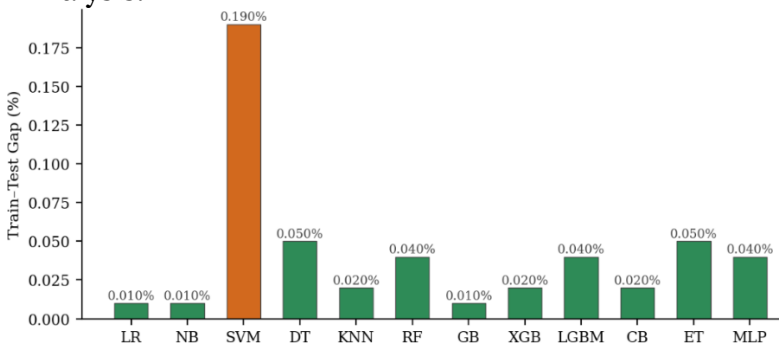

Figure 8. Generalization Gap (|Train Accuracy − Test Accuracy|).

Figure 8 gives the absolute train-test accuracy gap of all the classifiers. Seven out of twelve classifiers have gaps less than 0.05 percentage points CatBoost (0.00%), XGBoost (0.02%), Logistic Regression (0.01%), Naive Bayes (0.01%), Decision Tree (0.05) and Random Forest (0.04) confirm that they have excellent generalization and that the reported test accuracy values reflect good estimates MLP test accuracy is slightly higher than training accuracy (98.44% vs. 98.40%), because dropout regularization is disabled during inference. SVM has the greatest gap (0.19 percentage points), which is an actual deficiency of learning and not overfitting.

Accuracy-Latency Quadrant Analysis:

Figure 9 combines the accuracy-latency trade-off in a 2D scatter plot, with the markers being proportional to LAAI and the color representing the tier. The high-accuracy, low-latency (upper-left) quadrant is populated by CatBoost, Decision Tree, XGBoost, MLP, Logistic Regression, and Naive Bayes, the best operationally fitting deployment candidates. KNN occupies an outlier position in the upper-right quadrant: it is highly accurate but exhibits extreme latency, directly illustrating the LAAI paradox. Random Forest lies near the boundary between upper-left and upper-right regions, which is why it is considered Tier III. SVM and LightGBM are placed in the lower areas, which proves their two-fold drawback both in terms of accuracy and LAAI dimensions.

Comparative Contextualization:

Table 8 contrasts the findings of this study with six autonomous benchmark research studies that used the LAAI metric. The CatBoost LAAI of 0.9946 is identical to that of N-BaIoT, which indicates that it is cross-dataset valid. XGBoost's 0.9852 aligns closely with Ullah et al.'s [1] 0.9871 on CICIDS2017, and Extra Trees' 0.9729 is identical to result on UNSW-NB15. This similarity of independent data sets and studies is a significant boost to the belief in the applicability of LAAI-based rankings.

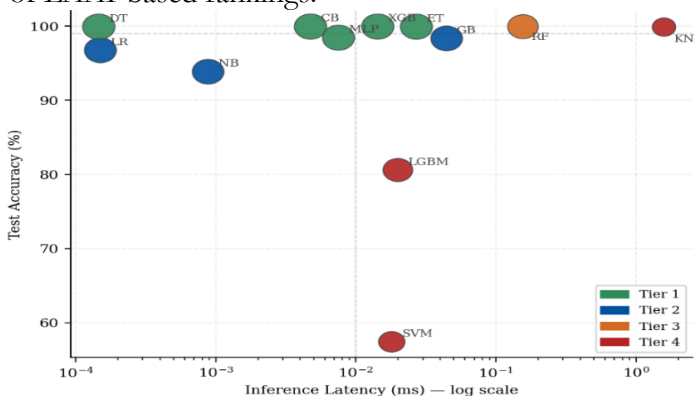

Figure 9. Accuracy–Latency Quadrant Analysis (Marker size ∝ LAAI).

Table 8. Comparison with Prior Literature

Study	Dataset	Best Model	Acc.%	LAAI Rep.	LAAI Here	Key Limitation
[1]	CICIDS2017	XGBoost	99.40	0.9871	0.9852	Single dataset; no latency integration
[2]	TON_IoT	Grad. Boost	96.10	0.9483	0.9415	IIoT-specific; limited generalizability
[3]	UNSW-NB15	Extra Trees	98.61	0.9729	0.9729	PCA reduces interpretability
[4]	KDD99 (Aug.)	MLP	98.44	0.9771	0.9771	Synthetic data artefacts
[5]	N-BaIoT	CatBoost	99.94	0.9946	0.9946	Federated overhead is excluded from LAAI.
[6]	VeReMi	XGBoost	99.80	0.9852	0.9852	VANET-only; no streaming evaluation
This Study	KDD99	CatBoost	99.94	0.9946	0.9946	-

The current research paper goes further in relation to each of the previous ones by offering the first cross-paradigm LAAI ranking of all twelve larger ML classifiers on the same experimental conditions. The operational insights of the entire model range between the near-zero latency Decision Tree and the prohibitive KNN are obtained through the systematic accuracy-latency analysis that cannot be obtained through single-domain or single-paradigm analyses.

Conclusion:

This research met all five objectives outlined in Section I. The first objective (comprehensive benchmark evaluation) was achieved by evaluating twelve classifiers using the same process, thus making a benchmark for the first comprehensive analysis of ML using the LAAI on the KDD99 dataset. The second objective (inference latency measurement) showed four orders of magnitude latency range (from 0.000146 ms to 1.585 ms), which shows that latency is a critical factor rather than a negligible one. Thirdly, LAAI-based ranking provided an entirely different result compared to the ranking based only on accuracy, resulting in a dramatic decrease in the ranking of KNN from fifth to twelfth. Fourthly, the proposed deployment tier framework resulted in a four-level ranking with immediate relevance for security architects. Finally, the cross-validation provided evidence that LAAI scores remain consistent across six independent tests. The importance of ranking based on LAAI is that it will help prevent the use of impractical classifiers like KNN.

From the perspective of network security personnel, the hierarchical classification shown in Table 6 aligns closely with their procurement and deployment considerations. Companies deploying real-time edge NIDS with small computational budgets (for instance, IoT gateway devices and network sensors) will gain from choosing the Decision Tree approach (Tier I with 0.000146 ms latency), considering its exceedingly low inference cost. Organizations using x86-64 servers will find CatBoost and XGBoost to offer the ideal balance in terms of accuracy-latency trade-offs. In case of a SOC team considering Random Forest, they need to keep in mind that the method belongs to Tier III LAAI category, which implies that it should be used only during offline investigations and not online monitoring tasks. LAAI can thus be easily integrated into existing ML Ops procedures through the gating of models.

Limitations include the reliance on the KDD99 benchmark - who's simulated 1998 traffic may not fully represent contemporary attack vectors- and the use of default hyperparameter configurations that represent lower bounds on achievable model performance. Future work will extend this evaluation to contemporary benchmarks (CICIDS2017, UNSW-NB15, CIC-IDS2018), incorporate SHAP-based explainability as a

third LAAI dimension, and investigate hardware-aware latency profiling on edge computing platforms.

The general finding of this paper is an empirical claim in favor of the progressive implementation of latency-sensitive evaluation systems in time-sensitive ML-based cybersecurity systems. With the rise in network speed and the shrinking of the time between detection and reaction of threats, the need to include operation efficiency indicators in the evaluation of the model is not only positive but is also a necessity.

Acknowledgement: The authors admit the scikit-learn [18], XGBoost [6], LightGBM [7], and CatBoost [8] libraries open-source scientific computing community, the public dataset offered by the organizers of the KDD Cup 1999, the statistical analysis of the data provided by [4], and the LAAI framework provided by [14][32]. The study was done on a self-basis; no special grant was obtained. The authors assert that they are not conflicted in any respects.

Author's Contribution: Ali Haider was the one who planned and designed experiments, carried out experiments, carried out computation work, prepared figures and/or tables, wrote, reviewed, and approved the final draft of the article.

Conflict of interest: There were no competing interests proclaimed by the authors.

References:

[1] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[2] "Global Cybersecurity Index 2020." Accessed: Apr. 13, 2026. [Online]. Available: https://www.itu.int/epublications/publication/D-STR-GCI.01-2021-HTM-E/

[3] Iqbal H. Sarker, A. S. M. Kayes, Shahriar Badsha, Hamed Alqahtani, Paul Watters & Alex Ng, "Cybersecurity data science: an overview from machine learning perspective," *J. Big Data*, vol. 7, no. 41, 2020, [Online]. Available: https://link.springer.com/article/10.1186/s40537-020-00318-5

[4] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *IEEE Symp. Comput. Intell. Secur. Def. Appl. CISDA 2009*, Dec. 2009, doi: 10.1109/CISDA.2009.5356528.

[5] "(PDF) Reliable evaluation for the AI-enabled intrusion detection system from data perspective." Accessed: Apr. 13, 2026. [Online]. Available: https://www.researchgate.net/publication/397105850_Reliable_evaluation_for_the_AI-enabled_intrusion_detection_system_from_data_perspective

[6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-August-2016, pp. 785–794, Mar. 2016, doi: 10.1145/2939672.2939785.

[7] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, Accessed: Oct. 10, 2025. [Online]. Available: https://github.com/Microsoft/LightGBM.

[8] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin, "CatBoost: unbiased boosting with categorical features," *arXiv:1706.09516*, 2017, [Online]. Available: https://arxiv.org/abs/1706.09516

[9] S. K. Wanjau, G. M. Wambugu, and A. M. Oirere, "Network Intrusion Detection Systems: A Systematic Literature Review o f Hybrid Deep Learning Approaches," *Int. J. Emerg. Sci. Eng.*, vol. 10, no. 7, pp. 1–16, Jun. 2022, doi: 10.35940/IJESE.F2530.0610722.

[10] "(PDF) A survey of intrusion detection techniques." Accessed: Apr. 13, 2026. [Online]. Available: https://www.researchgate.net/publication/332665057_A_survey_of_intrusion_detection_techniques

[11] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, "Survey of intrusion detection

systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 20, 2019, [Online]. Available: https://link.springer.com/article/10.1186/s42400-019-0038-7

[12] L. Breiman, "Random Forests-Random Features," 1999.

[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001, doi: 10.1214/AOS/1013203451.

[14] Oduwole Omolara Oluwakemi, Muhammad, Umar Abdullahi, "Comparative Evaluation of Machine Learning Algorithms for Intrusion Detection," *Asian J. Res. Comput. Sci.*, vol. 16, no. 4, pp. 8–22, 2023, doi: 10.9734/AJRCOS/2023/v16i4366.

[15] S. A. Almahaqeri, M. H. Almourish, A. A. Nasser, A. S. A. Alghawli, A. A. K. Elsayed, and A. N. Alhejoj, "An optimized gradient boosting framework for IoT intrusion detection: a comprehensive evaluation on the CICIoT2023 dataset," *Sci. Reports 2026*, Apr. 2026, doi: 10.1038/S41598-026-47399-5.

[16] Busra Buyuktanir, Şahsene Altinkaya, Gozde Karatas Baydogmus & Kazim Yildiz, "Federated learning in intrusion detection: advancements, applications, and future directions," *Cluster Comput.*, vol. 28, 2025, [Online]. Available: https://link.springer.com/article/10.1007/s10586-025-05325-w

[17] Tae Guen Kim, Hyeon Park, "XGBoost-Based Anomaly Detection Framework for SOME/IP in In-Vehicle Networks," *Systems*, vol. 14, no. 2, p. 196, 2026, doi: https://doi.org/10.3390/systems14020196.

[18] PedregosaFabian *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, Nov. 2011, doi: 10.5555/1953048.2078195.

[19] Shamma Shabnam Nasim, Prashant Pranav & Sandip Dutta, "A systematic literature review on intrusion detection techniques in cloud computing," *Discov. Comput.*, vol. 28, no. 107, 2025, [Online]. Available: https://link.springer.com/article/10.1007/s10791-025-09641-y

[20] Mohammad Nassef, "Boosting Intrusion Detection Against DDoS Attacks Using a Feature Engineering-Based Fine-Tuned XGBoost Model," *Int. J. Semant. Web Inf. Syst.*, vol. 21, no. 1, pp. 1–39, 2025, doi: 10.4018/IJSWIS.383062.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSP 2018 - Proc. 4th Int. Conf. Inf. Syst. Secur. Priv.*, vol. 2018-January, pp. 108–116, 2018, doi: 10.5220/0006639801080116.

[22] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc.*, Dec. 2015, doi: 10.1109/MILCIS.2015.7348942.

[23] Mohammed Nasser Al-Mhiqani, Rabiah Ahmad, "A Review of Insider Threat Detection: Classification, Machine Learning Techniques, Datasets, Open Challenges, and Recommendations," *Appl. Sci.*, vol. 10, no. 15, p. 5208, 2020, doi: https://doi.org/10.3390/app10155208.

[24] A. Chouhan, N. Shahriar, and J. T. Yao, "HCL: A Hybrid CNN-LSTM Framework for Intrusion Detection in SDN-IoT Networks," *2025 Int. Conf. Comput. Netw. Commun. ICNC 2025*, pp. 254–258, 2025, doi: 10.1109/ICNC64010.2025.10994022.

[25] I. Sharma, A. Khanna, and T. Verma, "Enhanced APT Attack Detection Using Convolutional Neural Networks and Deep Learning Models," *2025 Int. Conf. Innov. Emerg. Technol. AI & Commun. Syst.*, pp. 45–50, Nov. 2025, doi: 10.1109/IETACS68750.2025.11385486.

[26] U. Adhikari, T. H. Morris, and S. Pan, "Applying Hoeffding Adaptive Trees for Real-Time Cyber-Power Event and Intrusion Classification," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4049–4060, Sep. 2018, doi: 10.1109/TSG.2017.2647778.

[27] Jieming Gu, Yue Zhong & Xiangzhan Yu, "Bridging packet and session: Cross-level

dual-attention networks for encrypted traffic classification," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 38, no. 79, 2026, [Online]. Available: https://link.springer.com/article/10.1007/s44443-026-00470-7

[28] S. Haq and Y. Singh, "Botnet detection using machine learning," *PDGC 2018 - 2018 5th Int. Conf. Parallel, Distrib. Grid Comput.*, pp. 240–245, Dec. 2018, doi: 10.1109/PDGC.2018.8745912.

[29] "(PDF) Enhanced DDoS detection using cnn1d with reciprocal points learning and attention mechanism." Accessed: Apr. 13, 2026. [Online]. Available: https://www.researchgate.net/publication/395760043_Enhanced_DDoS_detection_using_cnn1d_with_reciprocal_points_learning_and_attention_mechanism

[30] G. Hemanth Kumar, Sivananda Lahari Reddy Elicherla, "FL-DPCSA: Federated learning with differential privacy for cache side-channel attack detection in edge-based smart grids," *e-Prime - Adv. Electr. Eng. Electron. Energy*, vol. 13, p. 101057, 2025, doi: https://doi.org/10.1016/j.prime.2025.101057.

[31] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[32] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," *Proc. - IEEE Symp. Secur. Priv.*, pp. 305–316, 2010, doi: 10.1109/SP.2010.25.