IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

AI-Driven Cybersecurity Threat Detection using Machine Learning for Real-Time Network Security

Shaikh Aman[1], Sayyad Rais[2], Shaikh Tofiq[3], Prof. Muskan[4], Dr. Vrushali Ranmalkar[5], Dr. P. M. Patil[6]
Department of Computer Application
Adsul Technical Campus, Chas, Ahilyanagar, India

Abstract: *Modern networks face a constantly shifting threat landscape shaped by ransomware, distributed denial-of-service (DDoS) campaigns, botnets, and stealthy zero-day intrusions. Static, signature-based defenses catch only what they already recognize, leaving a widening gap against novel attack behavior. This paper presents a machine-learning-driven threat detection pipeline designed to classify network flows as malicious or benign, further attributed to a specific attack category, in near real time. Eight supervised learning models—spanning classical classifiers, ensemble methods, and deep sequence learners—are trained and benchmarked on a labeled intrusion-detection dataset covering DDoS, port scanning, brute force, botnet, and web-based attacks. The pipeline combines systematic preprocessing, correlation-based feature selection, and stratified evaluation to produce a repeatable comparison across accuracy, precision, recall, F1-score, and detection latency. Results indicate that ensemble and deep-learning approaches deliver the strongest detection quality, while simpler models remain attractive where inference speed is the priority, pointing toward hybrid deployment strategies for production security operations. The reported figures reflect a single experimental protocol rather than a multi-seed statistical study, and this limitation, along with others, is discussed explicitly in Section XII.*

Keywords: network intrusion detection, machine learning, ensemble learning, deep learning, real-time security analytics, feature engineering, class imbalance.

I. INTRODUCTION

Enterprise networks now stretch across on-premises data centers, public cloud tenancies, and a growing population of IoT endpoints, multiplying the number of paths an attacker can exploit. Firewalls and signature-matching intrusion detection systems remain valuable for filtering known-bad traffic, but their reliance on pre-written rules means they are structurally blind to attacks that have not yet been catalogued. Zero-day exploits, polymorphic malware, and slow, low-and-slow reconnaissance are engineered specifically to slip past this kind of matching.

Machine learning reframes detection as a pattern-recognition problem rather than a rule-lookup problem. By learning statistical regularities that separate benign flows from malicious ones directly from traffic features—packet timing, flow duration, byte counts, protocol flags—a trained model can generalize to attack variants it has never explicitly seen, and can be retrained as new threat data becomes available. This adaptability is the central motivation for the framework developed in this paper.

This work makes three contributions. First, it defines an end-to-end pipeline that takes raw labeled flow data through cleaning, encoding, and feature selection before model training. Second, it evaluates eight representative algorithms—from interpretable linear models to deep recurrent networks—under an identical experimental protocol so that their trade-offs are directly comparable. Third, it discusses the practical implications of these trade-offs for deploying

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

detection models inside a real-time security operations workflow, where inference latency competes directly with detection accuracy for engineering priority. A fourth, cross-cutting contribution is an explicit accounting of the study's methodological limitations (Section XII), so that the reported comparisons are read with an accurate sense of their scope and confidence.

II. RELATED WORK

Intrusion detection research has followed the broader arc of machine learning itself. Early statistical and rule-mining approaches gave way to classical supervised classifiers—decision trees, Naive Bayes, and support vector machines—applied to handcrafted flow features. Ensemble techniques, particularly Random Forest and gradient-boosted trees such as XGBoost, subsequently demonstrated consistent accuracy gains by aggregating many weak learners, and they remain competitive baselines in current literature.

More recently, deep learning has been applied directly to traffic data. Convolutional architectures treat flow features or raw packet bytes as image-like tensors to extract local patterns, while recurrent architectures—especially Long Short-Term Memory (LSTM) networks—model the temporal ordering of packets within a session, which is useful for attacks that unfold over time, such as slow-rate DDoS or multi-stage lateral movement.

Benchmark datasets have played a large role in enabling comparable evaluation across studies. The CICIDS2017 dataset, released by the Canadian Institute for Cybersecurity, is widely used because it captures a broad mixture of modern attack types alongside benign background traffic, recorded with a documented feature-extraction methodology.

Reported open challenges across this literature include class imbalance between rare attack categories and abundant benign traffic, sensitivity of deep models to hyperparameter choices, and limited interpretability of black-box classifiers in security-critical settings. This paper adopts the same pipeline structure used elsewhere in the literature to address these challenges in principle, while being explicit in Section XII about which of them the present study fully resolves versus merely acknowledges.

III. PROBLEM STATEMENT

Signature-based detection systems are, by construction, only as good as the signature database behind them; an attack pattern that has not been added cannot be matched. This leaves organizations exposed during the window between a new exploit's first use and the point at which a signature is written, tested, and deployed—a window attackers actively exploit.

A learning-based alternative must satisfy several competing constraints simultaneously: it must generalize beyond the exact attacks in its training set, it must operate within the latency budget of live traffic inspection, and it must keep false positives low enough that security analysts are not overwhelmed with noise. This paper treats the selection and comparison of candidate models as a search for the best available trade-off among these constraints, rather than assuming any single algorithm is universally optimal.

IV. OBJECTIVES

- Build a reproducible preprocessing pipeline for labeled network-flow data suitable for supervised learning.
- Train and tune eight machine learning and deep learning classifiers under identical data splits.
- Quantify each model's detection quality using accuracy, precision, recall, and F1-score.
- Measure per-flow inference latency to assess real-time deployment feasibility.
- Identify the operating points where ensemble or deep models justify their added computational cost over simpler classifiers.
- Document, rather than obscure, the points at which the present protocol falls short of full reproducibility (Section XII), so future extensions of this work know precisely where to add rigor.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

V. RESEARCH METHODOLOGY

Phase 1: Data Acquisition

Labeled flow-level traffic is sourced from the CICIDS2017 benchmark, which includes benign traffic alongside DDoS, port scanning, botnet, brute-force, SQL injection, cross-site scripting, and Heartbleed exploitation samples captured over a multi-day period.

Phase 2: Preprocessing

- Dropping records with missing or infinite feature values.
- Removing exact-duplicate flow records.
- Scaling continuous features with min-max normalization.
- One-hot encoding categorical protocol and flag fields.
- Correlation-based feature selection to remove redundant, highly collinear features (a fixed |r| threshold was applied to the Pearson correlation matrix; the exact threshold and resulting feature count are implementation details that should be reported alongside any code release so the selection is exactly reproducible).
- Class-imbalance handling: CICIDS2017 is heavily skewed toward benign traffic, with categories such as Heartbleed represented by only a handful of flows. This pipeline treats imbalance primarily through stratified sampling at the train/test split stage and through class-weighted loss functions during training, rather than synthetic oversampling; this choice, and its effect on rare-class recall, is discussed as a limitation in Section XII.

Phase 3: Model Training

Each of the eight candidate algorithms is trained on an identical 80/20 stratified train-test split, preserving the relative frequency of each traffic class. The split is performed after deduplication and is stratified by attack label; because CICIDS2017 flows are captured in contiguous time windows, care is taken to avoid partitioning that would let near-duplicate flows from the same short time window appear in both the training and test partitions, which can otherwise inflate reported accuracy. Hyperparameters for tree-based and boosting models are tuned via grid search on a validation fold within the training partition; the neural architectures (ANN, LSTM) use early stopping on validation loss to avoid overfitting. Final hyperparameter values and network architectures used for each model are recorded in the project's experiment log so that results can be reproduced exactly; a summary of these settings belongs in an appendix or supplementary repository.

Phase 4: Evaluation

- Accuracy
- Precision
- Recall
- F1-score
- Per-flow inference latency, measured at single-flow (batch size = 1) inference on the hardware described in Section IX, to reflect the streaming, one-flow-at-a-time nature of live traffic inspection rather than batched throughput.

Phase 5: Comparative Analysis

Results across all models are tabulated side by side to identify which algorithm(s) offer the best balance of detection quality and speed for a real-time deployment scenario. Because several models finish within a fraction of a percentage point of one another (Table I), differences of less than roughly one percentage point should be read as practically comparable rather than as a strict ranking, absent variance estimates from repeated runs (see Section XII).

VI. PROPOSED DETECTION FRAMEWORK

The proposed real-time detection pipeline is organized as a linear sequence of stages: live network traffic capture; packet parsing and flow reconstruction; data cleaning and normalization; feature engineering and selection; the ML/DL

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

classification engine; threat scoring and categorization; and alert generation feeding a security-operations-center (SOC) dashboard.

Incoming traffic is captured and reconstructed into bidirectional flows, from which statistical features are computed. These features are cleaned and normalized before being passed to the trained classification engine, which scores each flow and forwards suspicious flows to an alerting layer feeding the SOC dashboard for analyst review. In a production deployment, the accuracy/latency trade-off documented in Section X motivates a two-tier version of this pipeline: a fast, lightweight model triages the full traffic volume, and only flows it flags as suspicious are passed to a heavier, higher-accuracy model for confirmation.

VII. DATASET DESCRIPTION

The CICIDS2017 dataset provides more than 80 statistical features per network flow, including flow duration, forward/backward packet counts, byte volumes, inter-arrival time statistics, and TCP flag counts. It labels traffic across the following categories, used in this study: DDoS, Port Scanning, Botnet, Brute Force (FTP/SSH), SQL Injection, Cross-Site Scripting (XSS), Infiltration/Web Attacks, and Heartbleed.

Class frequencies across these categories are highly unequal — benign traffic and DDoS/port-scan flows are comparatively abundant, while Heartbleed and SQL Injection samples number in the low hundreds or fewer across the full capture. This imbalance, exact counts for which should be reported per split in any reproducible release of this work, is the direct cause of the reduced recall observed for rare categories in Section X.

VIII. MACHINE LEARNING MODELS EVALUATED

Logistic Regression — A linear baseline that estimates class probability from a weighted sum of flow features; fast to train and easy to interpret, but limited on non-linear decision boundaries.

Decision Tree — Splits the feature space recursively into interpretable if-then rules; prone to overfitting without depth limits or pruning.

Random Forest — Aggregates many decorrelated decision trees via bagging, reducing variance and improving robustness to noisy features.

Support Vector Machine (SVM) — Finds a maximum-margin separating hyperplane, optionally in a kernel-transformed feature space, effective in high-dimensional settings.

K-Nearest Neighbors (KNN) — Classifies a flow by majority vote among its nearest neighbors in feature space; simple but sensitive to feature scaling and dataset size.

XGBoost — A regularized gradient-boosting framework that builds trees sequentially to correct prior errors, frequently achieving state-of-the-art tabular performance.

Artificial Neural Network (ANN) — A fully connected feed-forward network that learns non-linear feature interactions through multiple hidden layers.

Long Short-Term Memory (LSTM) — A recurrent architecture with gated memory cells that captures temporal dependencies across sequential packets within a flow.

IX. EXPERIMENTAL SETUP

Software Stack

- Python 3.11
- Jupyter Notebook / Google Colab
- Pandas, NumPy for data handling
- Scikit-learn for classical models
- TensorFlow/Keras for ANN and LSTM
- XGBoost library
- Matplotlib/Seaborn for visualization

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

Hardware

- Intel Core i7 (8th generation or later) processor
- 16 GB system RAM
- NVIDIA GPU (optional, used to accelerate ANN/LSTM training and, where used, latency measurement for those models)

Evaluation Protocol

An 80/20 stratified split is used, with five-fold cross-validation applied during hyperparameter tuning to reduce sensitivity to a single split. Reported test-set metrics in Table I correspond to the held-out 20% partition evaluated once after tuning; they are not averaged across cross-validation folds, and no confidence interval or standard deviation is reported around them. Readers should treat the ranking implied by close scores (e.g., Random Forest, XGBoost, ANN, and LSTM, all within roughly one percentage point of each other) as indicative rather than statistically confirmed, pending repeated-run variance estimates.

X. RESULTS AND DISCUSSION

Table I summarizes test-set performance across all eight models under the protocol described in Section IX. Ensemble methods (Random Forest, XGBoost) and the deep sequence model (LSTM) achieve the highest accuracy and F1-scores, consistent with their ability to model complex, non-linear relationships between flow features and attack labels. Logistic Regression trails the field, consistent with the expectation that a purely linear decision boundary under-fits the non-linear structure typically present in labeled attack traffic of this kind.

Detection latency tells a complementary story: Decision Tree and Logistic Regression return predictions fastest, while LSTM's sequential processing makes it the slowest of the eight, despite its top-tier accuracy. This inverse relationship between accuracy and latency is the central engineering trade-off facing any real-time deployment.

Key observations:

- XGBoost and Random Forest offer the best accuracy-to-latency ratio among the tested models.
- LSTM edges out the other models on raw detection quality, at the cost of the highest inference latency of the eight; the exact latency multiple should be reported directly from the measured per-flow timings rather than approximated.
- Logistic Regression remains useful as a lightweight first-pass filter ahead of a heavier secondary classifier.
- Class imbalance (e.g., comparatively rare Heartbleed and SQL Injection samples) noticeably lowers recall for underrepresented attack types across all models, which is consistent with those categories receiving little training signal relative to benign and DDoS/port-scan traffic.

XI. COMPARATIVE ANALYSIS

TABLE I. Model Performance Comparison on the Test Set

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	91.2	89.4	88.7	89.0
Decision Tree	94.6	93.8	93.1	93.4
Random Forest	98.3	98.0	97.9	97.9
SVM	95.7	95.1	94.6	94.8
KNN	93.5	92.9	90.8	91.8
XGBoost	98.6	98.4	98.2	98.3
ANN	97.9	97.5	97.2	97.3

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

LSTM	98.8	98.6	98.5	98.5

The margin separating the top four models (Random Forest, XGBoost, ANN, and LSTM) is under one percentage point on every metric, which is smaller than the run-to-run variance typically observed for neural models on flow-based intrusion datasets. Consequently, the practical conclusion supported by this table is that ensemble and deep-learning approaches occupy a materially higher-performing tier than the linear and instance-based baselines (Logistic Regression, KNN), rather than that any single model within that top tier is definitively best; the latency figures in Section X are what differentiate them for deployment purposes.

XII. LIMITATIONS

The following limitations apply to the methodology and results as presented, and should be addressed before the pipeline is treated as production-validated or before Table I is cited as a definitive ranking:

- Single-run evaluation: reported metrics reflect one train/test split rather than an average with variance across multiple seeds or cross-validation folds, so small differences between top-performing models are not statistically established.
- Unreported hyperparameters and architectures: the specific grid-search ranges, final hyperparameter values for tree-based/boosting models, and layer/unit/sequence-length configuration for the ANN and LSTM are not enumerated in this paper; without them, the comparison is not independently reproducible.
- Unspecified feature-selection threshold: the correlation cutoff used to drop redundant features, and the resulting number of retained features, are not stated.
- Imbalance handling is partial: stratified sampling and class weighting mitigate but do not eliminate the effect of rare-class scarcity (e.g., Heartbleed), and recall for those classes remains the weakest part of every model's performance.
- Binary-versus-multiclass framing: the abstract and problem statement describe benign/malicious classification, while the dataset and evaluation cover eight distinct attack categories; Table I should specify whether reported metrics are macro-averaged across classes, micro-averaged, or computed on a collapsed binary label, since this materially changes how the numbers should be read.
- Latency benchmarking conditions: per-flow latency depends heavily on batch size and on whether GPU acceleration is active; this should be stated explicitly per model rather than assumed constant.
- Dataset age and drift: CICIDS2017 reflects attack traffic patterns from its capture period and may not represent current adversary techniques, which is a limitation shared with most published work using this benchmark and is the motivation for the drift-robustness item in Future Work.

XIII. CONCLUSION

This paper demonstrates that machine-learning-based classifiers, evaluated under a common protocol, can detect a broad range of network attacks with substantially higher accuracy than static, signature-based methods. Ensemble learners (Random Forest, XGBoost) and deep sequence models (LSTM) lead on detection quality, while simpler models retain a role wherever inference speed is the binding constraint. A production-grade deployment would likely combine both: a fast model to triage the bulk of traffic and a heavier model reserved for flows flagged as suspicious. These conclusions hold at the level of tiers of models rather than a strict single-model ranking, given the limitations enumerated in Section XII.

XIV. FUTURE WORK

- Applying Explainable AI methods (SHAP, LIME) to make ensemble and deep-model decisions auditable for security analysts.
- Testing federated learning so multiple organizations can jointly improve detection models without sharing raw traffic data.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, October 2026
Impact Factor: 8.2

- Deploying the pipeline on a streaming platform (e.g., Apache Kafka) to validate true real-time throughput under production load.
- Evaluating model robustness against adversarially crafted traffic designed to evade the classifier.
- Compressing models (pruning, quantization) for deployment on resource-constrained edge and IoT gateways.
- Repeating evaluation across multiple random seeds/cross-validation folds and reporting variance, to place the top-tier model comparison in Section XI on firmer statistical footing.
- Publishing full hyperparameter grids, final configurations, and the feature-selection threshold alongside the code, so the pipeline in Section V is exactly reproducible by other researchers.

REFERENCES

1. D. E. Denning, "An intrusion-detection model," IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222–232, 1987.
2. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), 2018, pp. 108–116.
3. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794.
4. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
5. L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
6. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.
7. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in Proc. Military Communications and Information Systems Conf. (MilCIS), 2015, pp. 1–6.