Machine Learning Approaches for Network Intrusion Detection: A Comparative Study

Yunfei Quan, Peifeng Yao, Yulong Wang, Ze Xu, Shubo Wang

Department of Electrical and Computer Engineering
New York University Tandon School of Engineering
Brooklyn, NY 11201, USA

{yq2341, py2330, yw9562, zx2087, sw6878}@nyu.edu

December 2025

Abstract

Network intrusion detection is a critical component of modern cybersecurity infrastructure, requiring accurate and efficient classification of network traffic to identify potential threats. This study presents a comprehensive comparative analysis of machine learning approaches for network intrusion detection, evaluating seven different algorithms including Logistic Regression (L1/L2), Support Vector Machine, Random Forest, Neural Network, XGBoost, and LightGBM. Using a cleaned subset of the CSE-CIC-IDS2018 dataset containing 360,000 network flow records with 78 features across 9 traffic classes (1 benign and 8 attack types), we systematically compare model performance in terms of accuracy, precision, recall, and F1-score. Our experimental results demonstrate that LightGBM achieves the highest test accuracy of 91.27% with an F1-score of 0.9102, outperforming all other models. The study reveals that DoS attacks (SlowHTTPTest) and FTP-BruteForce exhibit similar network flow characteristics, leading to classification challenges. Feature importance analysis identifies TCP initial window size, destination port, and flow timing statistics as the most discriminative features for distinguishing between attack types. These findings offer practical guidance for deploying machine learning-based intrusion detection systems in production environments.

Keywords: Network Intrusion Detection, Machine Learning, Classification, Gradient Boosting, Cybersecurity

1 Introduction

The rapid expansion of computer networks and internet connectivity has fundamentally transformed modern society, enabling unprecedented levels of communication, commerce, and data exchange. However, this digital transformation has also introduced significant security challenges, as malicious actors continuously develop sophisticated techniques to exploit network vulnerabilities. Network intrusion detection systems (IDS) serve as a critical line of defense, monitoring network traffic to identify and respond to potential security threats in real-time.

Traditional intrusion detection approaches rely primarily on signature-based methods, which match network traffic patterns against databases of known attack signatures. While effective for detecting previously documented threats, these methods fundamentally struggle with zero-day attacks and novel intrusion techniques that lack established signatures. This limitation has motivated extensive research into machine learning-based approaches, which can learn complex patterns from network traffic data and potentially generalize to detect previously unseen attack variants.

The application of machine learning to intrusion detection presents several technical challenges. Network traffic data is inherently high-dimensional, with dozens of features characterizing each network flow including packet statistics, timing information, and protocol-specific attributes. The classification task itself is often multi-class in nature, requiring systems to distinguish between normal traffic and multiple distinct attack categories. Furthermore, practical deployment considerations demand models that balance classification accuracy with computational efficiency, as IDS must process high volumes of traffic with minimal latency.

This study addresses these challenges through a systematic comparative analysis of seven machine learning algorithms for network intrusion detection using the CSE-CIC-IDS2018 dataset. We evaluate both traditional methods including Logistic Regression and Support Vector Machines, as well as modern ensemble approaches including Random Forest, XGBoost, and LightGBM. Our analysis encompasses not only classification performance metrics but also practical considerations including model interpretability through feature importance analysis and detailed examination of attack-specific detection capabilities.

The primary contributions of this work include: (1) a comprehensive performance comparison of seven machine learning models on a realistic network intrusion dataset; (2) detailed analysis of per-attack-type classification performance revealing specific challenges in distinguishing certain attack categories; (3) feature importance analysis identifying the most discriminative network flow characteristics for each attack type; and (4) practical recommendations for deploying machine learning-based intrusion detection systems.

2 Related Work

Machine learning approaches for network intrusion detection have been extensively studied over the past two decades. Early work by Lee and Stolfo [1] established foundational frameworks for applying data mining techniques to intrusion detection, demonstrating the potential for learning-based approaches to complement traditional signature methods. The release of benchmark datasets including KDD Cup 99 and its refined successor NSL-KDD [2] enabled systematic comparison of different algorithms and drove significant progress in the field.

Support Vector Machines emerged as a popular choice for intrusion detection due to their strong theoretical foundations and effectiveness in high-dimensional spaces. Mukkamala et al. [3] demonstrated competitive performance using SVM classifiers, though computational complexity remained a concern for large-scale deployment. Ensemble methods, particularly Random Forests, gained attention for their ability to handle high-dimensional data while providing inherent feature importance measures [4].

The CSE-CIC-IDS2018 dataset, developed by the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE), represents a significant advancement in intrusion detection research [5]. This dataset was created from a realistic network environment with 420 client machines and 30 servers organized across 5 departments, attacked by 50 machines over 10 days. The dataset contains over 16 million instances with 80 features extracted using CICFlowMeter-V3, covering seven attack scenarios including Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and network infiltration.

More recently, gradient boosting methods including XGBoost [6] and LightGBM [7] have achieved state-of-the-art results across many classification tasks. These algorithms construct ensembles of decision trees through iterative optimization, with LightGBM introducing histogram-based techniques that significantly improve training efficiency. Several studies have applied these methods to intrusion detection with promising results [8]. Leevy and Khoshgoftaar [9] conducted a comprehensive survey of intrusion detection models based on CSE-CIC-IDS2018, finding that gradient boosting methods consistently achieve high performance across different attack types.

Deep learning approaches have also been explored for network intrusion detection. Multi-layer perceptrons and more complex architectures including convolutional and recurrent neural networks have demonstrated the ability to learn hierarchical feature representations from raw network data [10]. However, the computational requirements and reduced interpretability of deep learning models present challenges for practical deployment.

Our work builds upon this foundation by providing a systematic comparison across multiple algorithm families using a balanced subset of CSE-CIC-IDS2018, with particular attention to understanding which attack types are most challenging to detect and why.

3 Dataset and Problem Formulation

3.1 CSE-CIC-IDS2018 Dataset

The dataset used in this study is derived from the CSE-CIC-IDS2018 dataset, a comprehensive network intrusion detection benchmark created collaboratively by the Canadian Institute for Cybersecurity (CIC) and the Communications Security Establishment (CSE). The original dataset was generated from a realistic network environment simulating an organization with 5 departments, 420 client machines, and 30 servers. An attacking infrastructure of 50 machines executed various attack scenarios over a 10-day period, while normal user behavior was simulated to generate benign traffic.

Network traffic was captured and processed using CICFlowMeter-V3, which extracts 80 flow-based features from raw packet captures. These features characterize bidirectional network flows and include statistics on packet sizes, inter-arrival times, TCP flags, and protocol-specific attributes. The original dataset contains over 16 million instances but exhibits significant class imbalance, with benign traffic comprising approximately 83% of all samples.

3.2 Dataset Preparation

To facilitate model comparison and reduce computational requirements, we utilized a cleaned and balanced subset of the original dataset containing 360,000 samples with 40,000 samples per class. This balanced distribution eliminates the confounding effects of class imbalance, allowing direct comparison of algorithm performance across different attack types. Table 1 summarizes the key characteristics of the dataset used in this study.

Table 1: Dataset Statistics

Characteristic	**Value**
Total Samples	360,000
Number of Features	78
Number of Classes	9 (1 benign + 8 attacks)
Training Set Size	252,000 (70%)
Test Set Size	108,000 (30%)
Class Distribution	Balanced (40,000 each)
Missing Values	None
Feature Extraction Tool	CICFlowMeter-V3

3.3 Traffic Classes and Attack Types

The dataset comprises one benign class representing normal network traffic and eight distinct attack types representing various cyber threats. Table 2 provides detailed descriptions of each class and its security implications.

Table 2: Traffic Classes and Attack Type Descriptions

Label	Class Name	Description
0	Benign	Normal, non-malicious network traffic representing legitimate user activities such as web browsing, file transfers, and email communications.
1	Bot	Traffic generated by botnet-infected machines communicating with command-and-control servers. Botnets are networks of compromised computers used for coordinated attacks.
2	DDoS-HOIC	Distributed Denial of Service attack using High Orbit Ion Cannon (HOIC), a tool that floods targets with HTTP requests to overwhelm web servers.
3	DoS-GoldenEye	Denial of Service attack using GoldenEye tool, which exploits HTTP Keep-Alive connections to exhaust server resources.
4	DoS-Hulk	Denial of Service attack using Hulk tool, which generates unique HTTP requests to bypass caching mechanisms and overwhelm web servers.
5	DoS-SlowHTTPTest	Slow HTTP attack that maintains connections by sending partial requests slowly, exhausting server connection pools without requiring high bandwidth.
6	FTP-BruteForce	Brute force attack against FTP servers attempting to guess valid username/password combinations through systematic trial.
7	Infiltration	Stealthy attack where adversaries gain initial access and attempt to move laterally within the network while evading detection.
8	SSH-BruteForce	Brute force attack against SSH servers attempting to gain unauthorized remote access through credential guessing.

3.4 Feature Categories

The 78 features extracted by CICFlowMeter can be organized into several categories based on their semantic meaning. Table 3 summarizes these categories.

Table 3: Feature Categories

Category	Features
Basic Flow	Destination Port, Protocol, Flow Duration
Packet Counts	Total Forward/Backward Packets, Subflow Packets
Byte Statistics	Total Length Forward/Backward, Flow Bytes/s
Packet Length	Min/Max/Mean/Std of Forward and Backward Packet Lengths
Inter-Arrival Time	Flow IAT (Mean/Std/Max/Min), Forward/Backward IAT statistics
TCP Flags	FIN/SYN/RST/PSH/ACK/URG/CWE/ECE Flag Counts
Header Information	Forward/Backward Header Length
Flow Rates	Flow Packets/s, Forward/Backward Packets/s
Window Size	Initial Forward/Backward Window Bytes
Segment Size	Forward Segment Size Min/Avg, Backward Segment Size Avg
Active/Idle	Active/Idle Time Mean/Std/Max/Min

3.5 Problem Formulation

The intrusion detection task is formulated as a supervised multi-class classification problem. Given a feature vector $\mathbf{x} \in \mathbb{R}^{78}$ representing a network flow, the objective is to learn a function $f : \mathbb{R}^{78} \rightarrow \{0, 1, ..., 8\}$ that maps the input features to one of the 9 traffic classes (benign or one of 8 attack types). The classification function is learned from a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ where $N = 252,000$ and evaluated on a held-out test set of 108,000 samples.

Performance is evaluated using standard classification metrics including accuracy, precision, recall, and F1-score. Given the multi-class nature of the problem, we report weighted averages of precision, recall, and F1-score. Model robustness is assessed through 5-fold stratified cross-validation.

4 Methodology

4.1 Development Process

The development of our intrusion detection system followed a structured methodology consisting of ten milestones, as outlined in Table 4.

Table 4: Project Development Milestones

Milestone	Description
1	**Data Collection & Exploration**: Load the CSE-CIC-IDS2018 network traffic dataset, perform exploratory data analysis (EDA), examine data distributions, identify missing values, and understand the characteristics of features and attack class labels.
2	**Data Preprocessing**: Handle missing and infinite values, apply feature scaling using standardization, encode categorical labels, and split the dataset into training and test sets with stratified sampling.
3	**Feature Engineering**: Conduct PCA analysis to assess intrinsic dimensionality, evaluate feature importance using tree-based methods, and explore opportunities for dimensionality reduction.
4	**Model Development**: Implement seven machine learning models (Logistic Regression L1/L2, SVM, Random Forest, Neural Network, XGBoost, LightGBM) with appropriate hyperparameter configurations.
5	**Model Training**: Train all models on the training set (252,000 samples) using the preprocessed and scaled features.
6	**Model Evaluation**: Evaluate trained models on the test set (108,000 samples), compute performance metrics including accuracy, precision, recall, and F1-score, and compare results across all models.
7	**Hyperparameter Tuning**: Perform grid search and systematic parameter optimization to identify optimal configurations for each model.
8	**Cross-Validation**: Apply 5-fold stratified cross-validation to obtain robust performance estimates and assess model stability across different data partitions.
9	**Results Analysis**: Analyze confusion matrices, per-attack-type performance, feature importance rankings, and overfitting behavior; interpret findings in the context of network security.
10	**Report Writing**: Document methodology, experimental results, and findings; prepare comprehensive technical report with analysis and recommendations.

4.2 Data Preprocessing

Prior to model training, the dataset underwent several preprocessing steps to ensure data quality and compatibility with the selected algorithms. Initial inspection confirmed the absence of missing values, eliminating the need for imputation procedures. The dataset was checked for infinite values, which can arise from division operations in feature computation; none were detected in this dataset.

Feature scaling was performed using standardization, transforming each feature to have zero mean and unit variance:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j} \tag{1}$$

where μ_j and σ_j are the mean and standard deviation of feature j computed from the training set. Standardization is particularly important for algorithms sensitive to feature scales, including Logistic Regression, SVM, and Neural Networks.

The dataset was partitioned into training and test sets using stratified sampling to preserve class proportions, with 70% allocated to training (252,000 samples) and 30% reserved for final evaluation (108,000 samples).

4.3 Dimensionality Analysis

Principal Component Analysis (PCA) was applied to examine the intrinsic dimensionality of the feature space and assess potential for dimensionality reduction. The analysis reveals that approximately 23 principal components are required to capture 95% of the total variance, while 19 components suffice for 90%. The first two components alone explain only 34.09% of variance (20.56% and 13.53% respectively), indicating that the data exhibits complex structure not easily captured by linear projections. Table 5 summarizes the PCA results.

Table 5: PCA Variance Analysis

Metric	**Value**
Variance by PC1	20.56%
Variance by PC2	13.53%
Variance by PC1+PC2	34.09%
Components for 90% variance	19
Components for 95% variance	23

4.4 Machine Learning Models

We evaluated seven machine learning algorithms spanning linear models, kernel methods, ensemble methods, and neural networks. Each algorithm was configured with appropriate hyperparameters based on established best practices.

4.4.1 Logistic Regression

Logistic Regression models the posterior probability of class membership using the softmax function for multi-class classification. We evaluated two regularization variants. L2 regularization (Ridge) adds a penalty term $\lambda\|\mathbf{w}\|_2^2$ to the loss function, shrinking coefficients toward zero while retaining all features. L1 regularization (LASSO) uses penalty $\lambda\|\mathbf{w}\|_1$, which induces sparsity by driving some coefficients exactly to zero. Both variants used regularization strength $C = 1.0$ and maximum iterations of 1000.

4.4.2 Support Vector Machine

Support Vector Machines find the maximum-margin hyperplane separating classes in a transformed feature space. We employed the RBF (Radial Basis Function) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{2}$$

which enables non-linear decision boundaries through implicit mapping to high-dimensional space. The kernel coefficient γ was set to 'scale' (automatically computed based on feature variance).

4.4.3 Random Forest

Random Forest constructs an ensemble of decision trees, each trained on a bootstrap sample of the data with random feature subsets considered at each split. Final predictions are made by majority voting across trees. This approach reduces overfitting through averaging and provides natural estimates of feature importance. We configured 100 trees with maximum depth 20 and minimum samples for splitting set to 5.

4.4.4 Neural Network

A Multi-Layer Perceptron (MLP) with three hidden layers (128, 64, and 32 neurons) was implemented using ReLU activations. The network was trained using the Adam optimizer with initial learning rate 0.001, batch size 256, and L2 regularization coefficient 0.0001. Early stopping with patience of 10 epochs prevented overfitting by monitoring validation loss.

4.4.5 XGBoost

XGBoost implements gradient boosting with regularized objective functions and efficient tree construction algorithms. The model iteratively adds trees that minimize a differentiable loss function, with each tree fitted to the residuals of the current ensemble. We used 100 estimators with maximum depth 10, learning rate 0.1, and subsampling rate 0.8 for both samples and features.

4.4.6 LightGBM

LightGBM employs histogram-based algorithms and leaf-wise tree growth that significantly accelerate training compared to traditional gradient boosting implementations. The gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) techniques enable efficient handling of large-scale datasets. Configuration matched XGBoost with 100 estimators, maximum depth 10, and learning rate 0.1.

4.5 Evaluation Metrics

Model performance was assessed using four standard classification metrics. Accuracy measures the overall proportion of correct predictions. Precision quantifies the fraction of predicted positives that are true positives. Recall (sensitivity) measures the fraction of actual positives correctly identified. The F1-score provides the harmonic mean of precision and recall, balancing both concerns.

For multi-class problems, we report weighted averages where each class contributes proportionally to its support:

$$\text{Weighted-}F1 = \sum_{c=1}^{C} \frac{n_c}{N} \cdot F1_c \tag{3}$$

where n_c is the number of samples in class c and N is the total number of samples.

Model stability was evaluated through 5-fold stratified cross-validation, computing mean and standard deviation of metrics across folds to assess generalization performance.

5 Experimental Results

5.1 Overall Model Performance

Table 6 presents the comprehensive performance comparison across all seven evaluated models. The results reveal substantial performance differences between algorithm families, with gradient boosting methods achieving the highest accuracy scores.

Table 6: Model Performance Comparison on Test Set

Model	Train Acc	Test Acc	Precision	Recall	F1-Score
Logistic Reg (L2)	0.8515	0.8500	0.8562	0.8500	0.8456
Logistic Reg (L1)	0.8423	0.8410	0.8439	0.8410	0.8365
SVM (RBF)	0.8613	0.8599	0.8884	0.8599	0.8501
Random Forest	0.9064	0.8852	0.9039	0.8852	0.8797
Neural Network	0.8741	0.8726	0.8844	0.8726	0.8689
XGBoost	0.9015	0.8886	0.8957	0.8886	0.8865
LightGBM	**0.9175**	**0.9127**	**0.9243**	**0.9127**	**0.9102**

LightGBM achieved the highest test accuracy of 91.27%, substantially outperforming all other models. XGBoost and Random Forest followed with test accuracies of 88.86% and 88.52% respectively. The linear models (Logistic Regression variants) showed the lowest performance, with test accuracies around 84-85%, indicating that the classification problem requires non-linear decision boundaries that linear models cannot adequately capture. SVM with RBF kernel achieved 85.99% accuracy, demonstrating the benefit of non-linear transformations but not matching the performance of ensemble methods.

5.2 Model Ranking

Table 7 presents the final ranking of models based on test accuracy, along with their F1-scores.

Table 7: Model Ranking by Test Accuracy

Rank	Model	Test Accuracy	F1-Score
1	LightGBM	91.27%	0.9102
2	XGBoost	88.86%	0.8865
3	Random Forest	88.52%	0.8797
4	Neural Network	87.26%	0.8689
5	SVM (RBF)	85.99%	0.8501
6	Logistic Reg (L2)	85.00%	0.8456
7	Logistic Reg (L1)	84.10%	0.8365

5.3 Per-Attack-Type Performance Analysis

To understand model behavior at a finer granularity, we analyzed per-class recall rates for the best-performing model (LightGBM). Table 8 presents the recall for each attack type along with security implications.

Table 8: Per-Attack-Type Detection Performance (LightGBM)

Label	Attack Type	Recall	Difficulty	Security Implication
0	Benign	0.81	Moderate	19% false positive rate may trigger unnecessary alerts
1	Bot	1.00	Easy	Excellent botnet detection capability
2	DDoS-HOIC	1.00	Easy	High-volume attacks reliably detected
3	DoS-GoldenEye	1.00	Easy	HTTP-based DoS well-characterized
4	DoS-Hulk	1.00	Easy	Unique request patterns easily identified
5	DoS-SlowHTTPTest	0.54	Hard	Low-bandwidth attacks often missed
6	FTP-BruteForce	0.95	Moderate	Most credential attacks detected
7	Infiltration	0.92	Moderate	Stealthy attacks mostly detected
8	SSH-BruteForce	1.00	Easy	Clear authentication patterns identified

The analysis reveals significant variation in detection capability across attack types. Bot traffic, DDoS-HOIC, DoS-GoldenEye, DoS-Hulk, and SSH-BruteForce attacks are detected with perfect recall (100%), indicating these attacks produce highly distinctive network flow patterns. FTP-BruteForce (95%) and Infiltration (92%) attacks achieve strong but imperfect detection rates.

The most significant challenge is DoS-SlowHTTPTest, which achieves only 54% recall. This attack type employs a fundamentally different strategy from other DoS attacks: rather than flooding targets with high volumes of traffic, it maintains connections by sending partial requests slowly, exhausting server connection pools without generating the distinctive high-traffic patterns that characterize other DoS variants. This low-and-slow approach produces network flows that more closely resemble legitimate traffic, explaining the classification difficulty.

5.4 Attack Type Confusion Analysis

Table 9 examines the primary sources of misclassification for challenging attack types.

Table 9: Misclassification Analysis for Challenging Classes

True Class	Confused With	Rate	Explanation
DoS-SlowHTTPTest	FTP-BruteForce	46%	Both maintain long-duration connections with sporadic data transmission, producing similar flow timing statistics
Benign	Infiltration	19%	Infiltration attacks deliberately mimic normal traffic patterns to evade detection
Infiltration	Benign	8%	Stealthy lateral movement resembles legitimate internal traffic

The confusion between DoS-SlowHTTPTest and FTP-BruteForce is particularly notable. Both attack types involve maintaining persistent connections over extended periods with intermittent data transmission, resulting in similar inter-arrival time distributions and packet rate statistics. This finding suggests that distinguishing these attack types may require additional features beyond those captured by CICFlowMeter, such as application-layer protocol analysis or connection state tracking.

5.5 Cross-Validation Results

To assess model stability and generalization, we performed 5-fold stratified cross-validation using LightGBM (the best-performing model). Table 10 summarizes the results.

Table 10: 5-Fold Cross-Validation Results (LightGBM)

Metric	Mean	Std Dev
Accuracy	0.9133	$\pm$ 0.0007
Precision	0.9257	$\pm$ 0.0014
Recall	0.9133	$\pm$ 0.0007
F1-Score	0.9106	$\pm$ 0.0007

The extremely low standard deviations (less than 0.002 for all metrics) indicate highly stable performance across different data partitions, suggesting the model generalizes well and is not sensitive to the particular training/test split. Table 11 shows the detailed results for each fold.

Table 11: Cross-Validation Results by Fold

Fold	Accuracy	F1-Score
1	0.9141	0.9117
2	0.9123	0.9099
3	0.9140	0.9111
4	0.9128	0.9099
5	0.9134	0.9107

5.6 ROC-AUC Analysis

Table 12 presents the micro-averaged Area Under ROC Curve (AUC) for models supporting probability predictions. All evaluated models achieve excellent AUC scores above 0.99, indicating strong discriminative ability across different classification thresholds.

Table 12: ROC-AUC Scores (Micro-Averaged)

Model	AUC
LightGBM	0.9978
XGBoost	0.9965
Random Forest	0.9963
Neural Network	0.9955

5.7 Overfitting Analysis

Table 13 compares training and test accuracy for each model to assess potential overfitting. The gap (Δ) between training and test accuracy indicates the degree of overfitting.

Table 13: Overfitting Analysis: Train vs Test Accuracy

Model	**Train Acc**	**Test Acc**	**Gap (Δ)**
Logistic Reg (L2)	0.8515	0.8500	0.0015
Logistic Reg (L1)	0.8423	0.8410	0.0013
SVM (RBF)	0.8613	0.8599	0.0014
Neural Network	0.8741	0.8726	0.0015
LightGBM	0.9175	0.9127	0.0048
XGBoost	0.9015	0.8886	0.0129
Random Forest	0.9064	0.8852	0.0212

Linear models (Logistic Regression variants), SVM, and Neural Network show minimal generalization gap ($\Delta \approx 0.001-0.002$), indicating no overfitting. Random Forest exhibits the largest gap ($\Delta = 0.021$), suggesting slight overfitting despite its strong test performance. XGBoost ($\Delta = 0.013$) and LightGBM ($\Delta = 0.005$) show moderate to excellent generalization, with LightGBM achieving the best balance between training performance and generalization.

5.8 Feature Importance Analysis

Understanding which features drive classification decisions is crucial for model interpretability and provides insights into what network characteristics distinguish different attack types. Table 14 shows the top 10 most important features identified by Random Forest based on Gini importance.

Table 14: Top 10 Feature Importances (Random Forest)

Rank	Feature	Importance	Security Interpretation
1	Init Fwd Win Byts	0.0951	TCP window size reveals OS fingerprints and connection behavior patterns
2	Dst Port	0.0935	Target service identification (SSH:22, FTP:21, HTTP:80)
3	Fwd Seg Size Min	0.0933	Minimum segment size indicates attack tool characteristics
4	Flow IAT Mean	0.0522	Average inter-arrival time distinguishes automated vs human traffic
5	Fwd Header Len	0.0521	Header anomalies indicate crafted packets
6	Flow Pkts/s	0.0399	Packet rate separates DoS from normal traffic
7	Fwd Pkts/s	0.0392	Forward packet rate indicates request patterns
8	Bwd Pkts/s	0.0312	Response rate reveals server behavior under attack
9	Flow IAT Max	0.0309	Maximum gap time identifies connection persistence
10	Fwd IAT Tot	0.0287	Total timing reveals session duration patterns

The feature importance analysis reveals several key insights. The TCP initial window size (Init Fwd Win Byts) emerges as the most discriminative feature, likely because different operating systems and attack tools use characteristic window sizes. Destination port is highly important as it directly indicates the target service (SSH, FTP, HTTP), which correlates strongly with attack type. Timing-related features (Flow IAT Mean, Flow IAT Max) are crucial for distinguishing automated attack traffic from normal human-generated traffic, as attack tools typically produce more regular timing patterns.

6 Discussion

6.1 Model Selection Recommendations

Based on our experimental results, LightGBM emerges as the recommended choice for network intrusion detection in this context. Its 91.27% test accuracy represents a substantial improvement over alternatives, with a 2.4 percentage point advantage over the second-best model (XGBoost). The model also demonstrates excellent generalization with minimal overfitting and stable cross-validation performance. For production deployment, LightGBM offers additional advantages including fast inference time and relatively small model size.

For applications where model interpretability is paramount, Random Forest provides a compelling alternative. Despite slightly lower accuracy (88.52%), it offers intuitive feature importance measures and decision paths that can be examined to understand classification logic. This interpretability may be valuable in security operations centers where analysts need to understand why a particular flow was flagged to make informed response decisions.

Linear models, while significantly less accurate, offer the fastest prediction times and simplest deploy-

ment. For preliminary screening or resource-constrained edge devices, Logistic Regression with L2 regularization provides a reasonable baseline with 85.00% accuracy.

SVM with RBF kernel achieved moderate accuracy (85.99%) but has computational complexity of $O(n^2)$ to $O(n^3)$, which limits its applicability for real-time processing of high-volume network traffic.

6.2 Analysis of Challenging Attack Types

The low detection rate for DoS-SlowHTTPTest (54% recall) represents a significant security concern, as these "low-and-slow" attacks can effectively deny service while evading detection. The confusion with FTP-BruteForce traffic occurs because both attack types share similar temporal characteristics: long-duration connections with intermittent data transmission. Traditional flow-based features fail to capture the semantic differences between slow HTTP requests and failed FTP login attempts.

To improve detection of slow-rate attacks, several approaches could be considered. First, application-layer features such as HTTP request completeness, response codes, and request-response correlation could provide additional discriminative power. Second, connection state tracking could identify the characteristic pattern of slow attacks where connections remain in partially-completed states. Third, aggregate behavior analysis examining multiple flows from the same source might reveal attack campaigns that appear benign at the individual flow level.

The 19% false positive rate for benign traffic (81% recall) also warrants attention. In production environments with predominantly benign traffic, this could generate substantial alert volumes. Threshold tuning, two-stage classification, or human-in-the-loop verification for borderline cases could help manage this challenge.

6.3 Feature Engineering Insights

The prominence of TCP window size and timing features in the importance rankings suggests several directions for improving detection. The Init Fwd Win Byts feature effectively fingerprints different systems and tools, indicating that attack tools often use distinctive TCP stack configurations. Incorporating additional TCP/IP stack fingerprinting features could enhance model performance.

The importance of inter-arrival time statistics confirms that timing analysis is essential for distinguishing automated attacks from human traffic. More sophisticated timing features, such as entropy of inter-arrival times or autocorrelation measures, could potentially improve discrimination of attacks that attempt to mimic normal timing patterns.

6.4 Limitations

Several limitations of this study should be acknowledged. First, the balanced class distribution, while simplifying evaluation, does not reflect real-world scenarios where benign traffic vastly outnumbers attack traffic. Production deployments would likely require techniques to handle class imbalance, such as cost-sensitive learning or sampling strategies. Second, our evaluation uses a static dataset from 2018; real network environments exhibit concept drift as traffic patterns and attack techniques evolve over time. Models would require periodic retraining to maintain effectiveness. Third, the dataset's specific composition of attack types may not generalize to all network environments, and new attack variants not represented in the training data may evade detection. Fourth, the 54% recall for slow HTTP attacks indicates a significant blind spot that could be exploited by sophisticated adversaries.

6.5 Future Work

Several directions merit further investigation. Ensemble methods combining predictions from multiple models could potentially improve performance on challenging attack types, particularly if different models exhibit complementary strengths. Deep learning architectures including convolutional and recurrent networks could capture temporal dependencies and sequential patterns in network flows that current methods may miss. Adversarial robustness evaluation would assess model vulnerability to deliberately crafted evasion attacks, an important consideration given that sophisticated attackers may attempt to modify their traffic to evade ML-based detection. Finally, real-time deployment studies would evaluate practical considerations including latency, throughput, and resource utilization in production network environments.

7 Conclusion

This study presented a comprehensive comparative analysis of machine learning approaches for network intrusion detection, evaluating seven algorithms on the CSE-CIC-IDS2018 dataset containing 360,000 network flows across 9 traffic classes. Our experiments demonstrate that gradient boosting methods, particularly LightGBM, achieve superior classification performance with 91.27% test accuracy and F1-score of 0.9102. The model exhibits excellent generalization with cross-validation accuracy of 91.33% $\pm$ 0.07%, indicating stable and reliable performance.

The analysis of per-attack-type performance reveals that most attack categories can be detected with high accuracy, with Bot, DDoS-HOIC, DoS-GoldenEye, DoS-Hulk, and SSH-BruteForce achieving perfect recall. However, slow-rate DoS attacks (SlowHTTPTest) remain challenging to detect, achieving only 54% recall due to their similarity to legitimate long-duration connections. This finding highlights the need for complementary detection mechanisms for low-and-slow attacks.

Feature importance analysis reveals that TCP window size, destination port, and packet timing characteristics are the most discriminative features for intrusion detection. These findings provide actionable insights for feature engineering and suggest that attackers attempting to evade detection should focus on mimicking normal values for these critical features.

These findings contribute to the growing body of evidence supporting machine learning approaches for network security applications. While certain attack types remain challenging to detect through flow-based analysis alone, the overall performance levels achieved suggest that machine learning-based intrusion detection is viable for practical deployment as part of a defense-in-depth security strategy.

References

[1] Lee, W., & Stolfo, S. J. (2000). A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4), 227-261. https://doi.org/10.1145/382912.382914

[2] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1-6. https://doi.org/10.1109/CISDA.2009.5356528

[3] Mukkamala, S., Janoski, G., & Sung, A. (2002). Intrusion detection using neural networks and support vector machines. *Proceedings of the International Joint Conference on Neural Networks*, 1702-1707. https://doi.org/10.1109/IJCNN.2002.1007774

[4] Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(5), 649-659. https://doi.org/10.1109/TSMCC.2008.923876

[5] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *4th International Conference on Information Systems Security and Privacy (ICISSP)*, 108-116. https://www.unb.ca/cic/datasets/ids-2018.html

[6] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794. https://doi.org/10.1145/2939672.2939785

[7] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 3146-3154. https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree

[8] Dhaliwal, S. S., Nahid, A. A., & Abbas, R. (2018). Effective intrusion detection system using XGBoost. *Information*, 9(7), 149. https://doi.org/10.3390/info9070149

[9] Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data*, 7, 104. https://doi.org/10.1186/s40537-020-00382-x

[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/

[11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. https://jmlr.org/papers/v12/pedregosa11a.html

[12] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153-1176. https://doi.org/10.1109/COMST.2015.2494502