



Machine Learning-Based Network Intrusion Detection System for Secure Communication Networks

Faizaan Ekanayake

*Department of Computer Science and Engineering, Tonle Sap Institute of Engineering and Commerce,
Cambodia*

faizaan.ekanayake@tsiec-kh.org

Peer Review Information

*Submission: 01 Jan 2020
Revision: 16 Feb 2020
Acceptance: 26 Feb 2020*

Keywords

*Network Intrusion
Detection, Machine
Learning, Cybersecurity,
Secure Communication,
Anomaly Detection.*

Abstract

The rapid expansion of Internet services, cloud platforms, mobile systems, and machine-to-machine communication has increased the attack surface of modern communication networks. Signature-based intrusion detection remains useful for previously observed threats, but its dependence on predefined rules limits its ability to detect new or rapidly evolving attacks. This methodology paper presents a machine learning-based network intrusion detection system (ML-NIDS) for secure communication networks. The proposed workflow converts raw network flows into a structured feature space, removes redundant and noisy attributes, addresses class imbalance, trains multiple supervised learning algorithms, and selects a final classifier using accuracy, precision, recall, F1-score, false-positive rate, and inference latency. The methodology is designed around public benchmark data such as UNSW-NB15 and CICIDS2017, whose traffic profiles contain normal sessions and contemporary attack families. Random Forest, Support Vector Machine, k-Nearest Neighbour, XGBoost-style gradient boosting, and a multilayer perceptron are considered as candidate models. The design also incorporates stratified train-test separation, cross-validation, normalization, feature ranking, and confusion-matrix analysis to reduce optimistic bias. A real-time deployment layer converts trained predictions into alerts that can support access control, traffic filtering, and incident response. Illustrative benchmark-style results show that ensemble tree models can achieve a favorable balance between detection accuracy and processing overhead. The methodology provides a reproducible foundation for evaluating ML-based intrusion detection before implementation in operational secure communication networks.

Introduction

Communication networks now carry business transactions, healthcare data, industrial control messages, multimedia streams, and authentication traffic. This dependence makes confidentiality, integrity, and availability central requirements of network design. Firewalls and cryptographic protocols protect important boundaries and communication channels, but they cannot independently identify every

malicious behavior occurring inside permitted sessions. Network intrusion detection systems (NIDSs) therefore provide a complementary monitoring layer that observes traffic patterns and identifies events associated with probing, denial-of-service, privilege abuse, malware communication, or unauthorized access. Conventional misuse-based systems are effective when an attack signature is known; however,

their detection capability decreases when attackers alter payloads, timing patterns, or protocol behavior.

Machine learning (ML) provides an alternative in which decision boundaries are inferred from examples rather than manually encoded. Sommer and Paxson (2010) emphasized that ML-based intrusion detection must be evaluated carefully because laboratory traffic is usually cleaner and more stationary than real operational networks. Dataset realism is therefore a key methodological concern. Shiravi et al. (2012) argued for systematic generation of benchmark datasets that preserve realistic traffic characteristics, while Moustafa and Slay (2015) introduced UNSW-NB15 to better represent modern normal and malicious network activities. Later, Sharafaldin et al. (2018) proposed CICIDS2017 with benign traffic and several common attack families. These developments make it possible to evaluate ML-NIDS models under more representative conditions.

The objective of this study is to design and evaluate a reproducible ML-based intrusion detection methodology for secure communication networks. Rather than assuming that one classifier is universally optimal, the framework compares statistical, distance-based, margin-based, ensemble, and neural learning approaches under the same preprocessing and evaluation conditions. The study focuses on both predictive performance and deployability, because a high-accuracy model may still be unsuitable for real-time use if its false alarm rate or inference cost is excessive.

Literature Review

The application of data mining and ML to cybersecurity has developed from conventional classifiers toward ensemble and deep learning models. Buczak and Guven (2016) reviewed decision trees, support vector machines, Bayesian methods, clustering, artificial neural networks, and related techniques used for cyber intrusion detection. Their survey showed that model performance depends strongly on data representation, attack distribution, feature selection, and evaluation design rather than on

classifier choice alone. Aburomman and Reaz (2017) further showed that ensemble and hybrid classifiers are increasingly important for strengthening intrusion classification through complementary learners and feature-processing stages. Ahmed et al. (2016) similarly organized network anomaly detection methods and highlighted the need to relate anomaly definitions to attack behavior and dataset characteristics.

A major limitation of early intrusion detection research was its dependence on aging or synthetic datasets. Moustafa and Slay (2015) created the UNSW-NB15 dataset by combining real normal traffic with contemporary synthetic attacks and extracting 49 network features. Sharafaldin et al. (2018) later developed CICIDS2017 to provide more diverse traffic and attack categories. These datasets help address the concerns raised by Sommer and Paxson (2010), although they still cannot perfectly reproduce the changing distribution, encryption, and application diversity of a production network.

Deep learning studies further demonstrated the value of learned feature representations. Javaid et al. (2016) applied self-taught learning with a sparse autoencoder for NIDS classification, showing that representation learning can support flexible detection. Vinayakumar et al. (2019) compared deep neural networks with conventional ML classifiers across several intrusion datasets and reported the importance of dataset-dependent tuning. Liu and Lang (2019) reviewed ML and deep learning approaches and identified challenges related to imbalanced classes, feature dimensionality, computational overhead, concept drift, and interpretability. Gamage and Samarabandu (2020) conducted a controlled survey and comparison of deep learning methods and found that model rankings can change with datasets and evaluation settings. Together, these studies indicate that a reliable methodology should include multiple models, realistic data, consistent preprocessing, class-aware metrics, and deployment-oriented evaluation.

Table 1. Selected Literature Supporting the Proposed Methodology

Study	Primary Contribution	Methodological Relevance
Sommer & Paxson (2010)	Critique of ML evaluation for operational NIDS	Supports realistic validation and cautious interpretation
Moustafa & Slay (2015)	UNSW-NB15 modern intrusion dataset	Provides contemporary benchmark traffic

Buczak & Guven (2016)	Survey of ML/data-mining techniques	Supports multi-algorithm comparison
Sharafaldin et al. (2018)	CICIDS2017 generation and traffic characterization	Supports recent benign/attack flow evaluation
Vinayakumar et al. (2019)	DNN and classical classifier benchmarking	Motivates baseline and neural comparisons
Gamage & Samarabandu (2020)	Objective comparison of deep learning NIDS methods	Supports controlled, metric-driven evaluation

Existing studies demonstrate high detection accuracy on benchmark datasets, but three methodological gaps remain important. First, many studies report results from a single classifier without a consistent baseline, making it difficult to determine whether performance gains arise from the learning algorithm or from preprocessing choices. Second, overall accuracy can conceal poor detection of minority attacks and can remain high even when false-positive rates are operationally unacceptable. Third, many experimental papers stop at offline classification and do not specify how a trained model would be integrated into a real-time secure communication environment. The present methodology addresses these gaps by applying one controlled pipeline to several ML families, evaluating class-aware and latency-oriented metrics, and defining an alert-oriented deployment stage. The specific objectives are: (1) to construct a clean and reproducible network-flow dataset from contemporary public intrusion benchmarks; (2) to identify informative features while avoiding data leakage; (3) to compare conventional and ensemble ML classifiers under identical train-test conditions; (4) to measure predictive performance using accuracy, precision, recall, F1-score, FPR, ROC-AUC, and inference latency; (5) to examine whether the selected model remains stable under class imbalance and threshold changes; and (6) to define a practical real-time decision path in which model outputs support secure communication controls. Two hypotheses guide the evaluation. H1 states that ensemble and nonlinear ML models will achieve significantly better F1-score than simple distance-based baselines on modern network-flow datasets. H2 states that feature selection and class-aware training will reduce false-positive and false-negative errors without materially increasing inference latency. These hypotheses are tested through cross-validated model comparison rather than through a single train-test result.

Methodology

The methodology follows a quantitative experimental design in which labeled network traffic is transformed into flow-level observations, processed using a fixed pipeline, and evaluated using repeated model training and testing. The workflow is designed so that each algorithm receives the same training and test information, enabling a fair comparison. The primary task is binary classification (normal versus attack), while the same pipeline can be extended to multiclass attack identification.

1. Threat Model and System Assumptions

The proposed system assumes an enterprise or institutional communication network in which flow metadata can be collected at a gateway, virtual switch, router, or security monitoring sensor. The adversary may conduct reconnaissance, brute-force login attempts, denial-of-service activity, bot communication, web attacks, exploitation, or unauthorized lateral movement. The IDS do not assume access to decrypted application payloads; instead, it uses statistical and protocol-level flow characteristics. The trusted computing base consists of the flow collector, preprocessing service, trained model repository, and alert channel. An attacker who fully compromises these components is outside the scope of the experiment, although model poisoning and adversarial evasion are recognized as future research problems.

To preserve validity, all transformations learned from data—including scaling parameters, feature rankings, oversampling ratios, and decision thresholds—must be estimated using training data only. This requirement prevents test information from influencing the model and is particularly important when datasets contain temporally or structurally similar flows. Where timestamps are available, an additional chronological holdout test is recommended to approximate deployment on future traffic.

Step 1: Dataset Selection and Traffic Representation

UNSW-NB15 and CICIDS2017 are selected because both include benign traffic and multiple attack categories and were created to improve upon older KDD-era benchmarks (Moustafa & Slay, 2015; Sharafaldin et al., 2018). Each record is represented as a flow rather than as a raw packet so that the model can learn properties such as duration, packet counts, byte counts, protocol, connection state, inter-arrival statistics, and directional flow behavior. Personally identifiable payload content is not required for the proposed classifier.

Step 2: Data Cleaning and Preprocessing

Duplicate records are removed and non-numeric fields are encoded. Missing and infinite values are replaced only when the replacement can be justified from the corresponding feature definition; otherwise, affected records are excluded. Highly skewed continuous variables are log-transformed where appropriate. Standardization is applied to SVM, kNN, and neural-network inputs, while tree-based classifiers are trained on the same cleaned but not necessarily standardized feature values. The dataset is divided using stratified sampling so that minority attack classes remain represented.

Step 3: Feature Selection and Class Balancing

Feature ranking combines mutual information, tree-based importance, and correlation screening. Redundant features with very high pairwise correlation are removed when they provide little additional information. To reduce bias toward majority benign traffic, the training partition can use class weights or controlled oversampling; the untouched test partition must retain the original distribution. This separation prevents information leakage and provides a more realistic estimate of false alarms and detection performance.

Step 4: Model Training and Validation

Five candidate classifiers are evaluated: Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbour (kNN), gradient-boosted decision trees (XGBoost-style), and a multilayer perceptron (MLP). These algorithms represent complementary learning mechanisms widely discussed in the intrusion-detection literature

(Buczak & Guven, 2016; Liu & Lang, 2019). Hyperparameters are selected using stratified five-fold cross-validation on the training set. The test set is used only once after model selection. For reproducibility, random seeds, software versions, feature lists, and split indices should be retained.

Step 5: Algorithm-Specific Training Strategy

Random Forest is trained with a controlled number of trees, bootstrap sampling, and class-weighted splitting to provide a robust nonlinear baseline. SVM is evaluated with radial-basis and linear kernels; the regularization parameter and kernel width are tuned through cross-validation. kNN is included as an interpretable distance-based baseline, with the number of neighbours selected from a small odd-valued grid. Gradient boosting sequentially fits shallow trees to classification residuals and is tuned for learning rate, tree depth, number of estimators, and subsampling. The MLP contains two or three hidden layers with ReLU activation, dropout or L2 regularization, early stopping, and a sigmoid or softmax output depending on whether the task is binary or multiclass.

Hyperparameter selection uses only cross-validation folds from the training partition. The optimization objective is primarily macro-F1 rather than accuracy so that minority attacks contribute to model selection. When computational resources allow, repeated stratified cross-validation is recommended to quantify variance across folds. The final selected configuration is retrained on the full training partition and then evaluated on the untouched test set.

Step 6: Performance Metrics and Statistical Comparison

The main metrics are accuracy, precision, recall (detection rate), F1-score, false-positive rate (FPR), and inference time. For true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$, $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$, $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$, $\text{F1} = 2(\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$, and $\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$. In operational NIDS deployment, recall and FPR are especially important because missed attacks reduce protection while excessive false alarms overload analysts.



Figure 1. Proposed Machine Learning-Based Intrusion Detection Workflow

Step 7: Real-Time Detection and Security Response

The selected model is exported into a real-time inference service connected to a flow collector. Incoming sessions are transformed using the same preprocessing parameters used during training. A probability or decision score is then compared with a configurable threshold. Low-risk traffic is forwarded normally, while suspicious traffic generates an alert containing the flow identifier, predicted class, confidence score, and relevant feature evidence. In a secure communication network, this alert can support rate limiting, temporary blocking, session re-authentication, network segmentation, or manual investigation. The IDS should operate as a decision-support control and should not automatically terminate critical services without policy safeguards.

2. Robustness, Generalization and Reproducibility

A robust NIDS should not be judged only on one random split. Three additional checks are

proposed. First, threshold-sensitivity analysis records the trade-off between recall and FPR across decision thresholds. Second, cross-dataset testing trains on one benchmark and evaluates compatible features on another dataset where feasible; a large performance collapse indicates dataset-specific learning. Third, model stability is examined by repeating training with multiple random seeds and reporting mean and standard deviation. These checks reduce the risk of selecting a model because of a favorable split rather than because of genuine generalization.

Reproducibility requires preserving preprocessing scripts, feature definitions, model versions, parameter grids, random seeds, and evaluation code. All reported metrics should be generated from a single immutable test set, and any later tuning should trigger creation of a new validation protocol rather than reuse of the same test labels. This practice aligns the security experiment more closely with reliable ML evaluation and reduces hidden researcher degrees of freedom.

Table 2. Proposed Experimental Configuration

Component	Configuration
Datasets	UNSW-NB15 and/or CICIDS2017; binary and optional multiclass labels
Train/Test Design	80:20 stratified split; 5-fold cross-validation on training data
Candidate Models	RF, SVM, kNN, gradient-boosted trees, MLP
Feature Processing	Encoding, cleaning, scaling where required, correlation screening, importance ranking
Imbalance Control	Class weighting or training-only oversampling
Metrics	Accuracy, precision, recall, F1, FPR, ROC-AUC, inference latency
Deployment Target	Flow-based real-time monitoring service with alert generation

Results

Because this paper defines a methodology rather than reporting a newly executed institutional experiment, the numerical values in Table 3 are illustrative benchmark-style results used to demonstrate how the proposed evaluation

should be reported. They must be replaced with actual experimental outputs when the pipeline is executed. The example assumes a binary intrusion-detection task with identical train/test partitions and tuned hyperparameters for every candidate model.

Table 3. Illustrative Comparative Results for the Proposed ML-NIDS Evaluation

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	FPR (%)	Latency (ms/flow)
kNN	96.2	95.4	95.9	95.6	3.7	0.42
SVM	96.9	96.5	96.1	96.3	3.0	0.55
MLP	98.1	97.8	97.9	97.8	1.9	0.39
Random Forest	98.4	98.1	98.0	98.0	1.5	0.24
Gradient Boosting	98.8	98.5	98.4	98.4	1.2	0.31

The illustrative comparison suggests that ensemble tree models can offer a strong balance between detection capability and execution speed. Gradient boosting produces the highest example accuracy and F1-score, while Random Forest offers slightly lower predictive performance with the lowest illustrated latency. SVM and kNN remain useful baselines but may become more expensive as the number of flows or training observations increases. The MLP is competitive, consistent with prior findings that neural approaches can learn complex traffic relationships, but its performance depends strongly on scaling, architecture, regularization, and training duration (Javaid et al., 2016;

Vinayakumar et al., 2019). A complete evaluation should go beyond a single aggregate accuracy value. First, per-attack recall should be reported because minority attacks may be hidden by majority performance. Second, confusion matrices should identify which attacks are confused with benign traffic or with one another. Third, the threshold should be tuned according to the operational cost of false alarms versus missed intrusions. Finally, performance should be tested under traffic drift or on a second dataset to estimate generalization. These requirements respond directly to concerns about closed-world evaluation in ML-based intrusion detection (Sommer & Paxson, 2010).

Table 4. Illustrative Confusion-Matrix Summary for the Selected Ensemble Model

Measure	Illustrative Count	Interpretation
True Positive (TP)	9,840	Attack flows correctly detected
True Negative (TN)	10,120	Benign flows correctly accepted
False Positive (FP)	155	Benign flows incorrectly alerted
False Negative (FN)	160	Attack flows missed by the detector

The illustrative confusion-matrix profile demonstrates why FPR and false negatives must be reported together. A model that misses few attacks but produces thousands of benign alerts can be unusable in a security operations centre, while a very conservative threshold may suppress alerts at the cost of undetected attacks. Accordingly, the deployment threshold should be selected using the organization's incident-handling capacity and the consequences of different attack types. High-risk attack categories may justify lower thresholds than low-impact anomalies. Resource consumption should also be measured during implementation. Training time affects model refresh frequency, whereas inference latency and memory footprint determine whether the classifier can keep pace with peak flow rates. For high-throughput

networks, batch inference, feature caching, parallel flow extraction, or a two-stage architecture can be used: a lightweight model filters clearly benign traffic and a more complex classifier evaluates ambiguous flows. This layered design can improve scalability without discarding the benefits of ML-based anomaly detection.

Conclusion and Discussion

This methodology paper presented a structured framework for developing and evaluating a machine learning-based network intrusion detection system for secure communication networks. The framework begins with realistic flow-based benchmark data, applies controlled cleaning and feature processing, handles class imbalance only within the training partition, and

evaluates multiple learning algorithms using a consistent validation design. It then extends the offline classifier into a real-time decision layer capable of producing security alerts and supporting defensive controls. The design reflects the evolution of NIDS research from conventional anomaly models toward ensemble and deep learning methods while retaining the caution that laboratory accuracy does not automatically translate into operational security. The literature indicates that data quality, attack diversity, feature representation, and evaluation methodology are at least as important as the classifier itself. Modern datasets such as UNSW-NB15 and CICIDS2017 improve realism, but no static benchmark can capture every encrypted application, zero-day behavior, topology change, or adversarial manipulation. Therefore, an operational implementation should periodically retrain or recalibrate models, monitor concept drift, maintain an auditable feature pipeline, and integrate ML alerts with human analyst review. Explainability is also important: security teams require evidence about why a flow was flagged, particularly when automated actions may interrupt legitimate communication. Future work can extend the methodology to multiclass attack recognition, online and federated learning, adversarially robust models, encrypted-traffic features, and hybrid architectures that combine ML anomaly detection with signature-based rules. Cross-dataset validation should be used to determine whether a model has learned general attack behavior rather than dataset-specific artifacts. With these safeguards, ML-based NIDS can provide a valuable adaptive layer for strengthening secure communication networks.

References

- Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. 2010 IEEE Symposium on Security and Privacy, 305-316. DOI: 10.1109/SP.2010.25.
- Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3), 357-374. DOI: 10.1016/j.cose.2011.12.012.
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. 2015 Military Communications and Information Systems Conference (MilCIS), 1-6. DOI: 10.1109/MilCIS.2015.7348942.
- 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. DOI: 10.1109/COMST.2015.2494502.
- Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, 19-31. DOI: 10.1016/j.jnca.2015.11.016.
- Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. *EAI Endorsed Transactions on Security and Safety*, 3(9), e2. DOI: 10.4108/eai.3-12-2015.2262516.
- Aburomman, A. A., & Reaz, M. B. I. (2017). A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Computers & Security*, 65, 135-152. DOI: 10.1016/j.cose.2016.11.004.
- Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 108-116. DOI: 10.5220/0006639801080116.
- Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20), 4396. DOI: 10.3390/app9204396.
- Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7, 41525-41550. DOI: 10.1109/ACCESS.2019.2895334.
- Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169, 102767. DOI: 10.1016/j.jnca.2020.102767.