

# Ensemble Machine Learning for Network Intrusion Detection: A Hybrid Stacking Framework with Optimized Feature Selection

Rafil Mohammed Jameel

University of Information Technology and Communications (UoITC)  
Baghdad, Iraq

Date of Submission: 01-05-2026

Date of Acceptance: 09-05-2026

## Abstract

Network Intrusion Detection Systems (IDS) must contend with heavy class imbalance, changing attack patterns, and heterogeneous network settings, making generalization and deployment extremely challenging. This paper introduces EnsembleShield++, a refined stacking-based ensemble model delivering strong cross-dataset performance and enhanced explainability. The framework combines Random Forest, XGBoost, and a Multilayer Perceptron as heterogeneous base learners, integrated through a regularized Logistic Regression meta-learner trained on out-of-fold predictions to prevent information leakage. A hybrid feature selection pipeline using mutual information filtering and iterative SHAP-informed elimination addresses feature redundancy while enhancing model transparency. Cross-dataset experiments (train-on-one, test-on-another) demonstrate robustness to distribution shift. Fold-aware SMOTE-ENN handles class imbalance without contaminating validation data. Comprehensive experiments on NSL-KDD, CICIDS2017, and UNSW-NB15 demonstrate that EnsembleShield++ achieves macro-F1 scores of 98.6%, 97.4%, and 96.8% respectively, with statistically significant improvements over all baselines (Wilcoxon,  $p < 0.05$ , 10 repeated runs). Class-specific SHAP analysis provides actionable feature interpretations for security practitioners.

**Keywords:** *Intrusion Detection System; Ensemble Learning; XGBoost; Random Forest; Stacking; SHAP; Class Imbalance; NSL-KDD; CICIDS2017; UNSW-NB15; Cross-Dataset Generalization*

The proliferation of internet-connected devices, cloud infrastructure, and remote work environments has dramatically expanded the attack surface available to malicious actors. According to recent industry estimates, the global economic cost of cybercrime is projected to exceed \$10 trillion annually by 2025, with network intrusions constituting a significant proportion of successful breaches [1]. At the heart of modern cyber defense lies the Intrusion Detection System (IDS)—a component designed to monitor network traffic, identify anomalous or policy-violating behavior, and trigger appropriate protective responses. Despite decades of research, designing an IDS that is simultaneously accurate, efficient, generalizable, and interpretable remains an open problem.

Traditional IDS methods can be broadly divided into two paradigms: signature-based detection and anomaly-based detection. Signature based systems are very good at detecting known threats but are not able to detect zero-day exploits. Systems that are based on anomalies provide more flexibility but are more likely to have a high rate of false positives below legitimate behavioral drift. Machine learning has become a promising backbone to anomaly-based IDS, but individual classifiers in most cases do not perform robustly and well balanced across all the attack types. at the same time, especially in case of uncommon but dangerous types of attack.

Ensemble learning addresses this limitation through the principled aggregation of multiple base learners. Stacking, a meta-learning generalization, allows a second-level model to learn how to optimally combine heterogeneous base learner predictions, potentially capturing synergies that

## I. Introduction

fixed aggregation rules cannot. Despite this theoretical appeal, practical challenges remain: high feature dimensionality, severe class imbalance, limited cross-dataset evaluation, and inadequate interpretability mechanisms.

### 1.1 Problem Statement

There are three research gaps that inspire this work. First, stacking-based ensemble architectures are under-investigated with fundamentally different base learners (gradient boosting, bagging and neural networks) in the context of IDS. Second, simple filter techniques are often used when selecting features in IDS, where the rich structure of network flow features is disregarded. Third, the benchmark of most IDS is tested on a single dataset and questions of cross-dataset generalization are not addressed. EnsembleShield++ will help to fill all three gaps.

### 1.2 Contributions

This work makes the following seven contributions:

1. We design EnsembleShield++, a heterogeneous stacking ensemble combining RF, XGBoost, and MLP via a leakage-safe Logistic Regression meta-classifier.
2. We propose a two-stage feature selection pipeline using mutual information filtering followed by iterative SHAP-based pruning for efficient selection and interpretability.
3. We propose a fold-aware SMOTE-ENN approach to class imbalance handling, preventing synthetic data leakage into validation partitions.
4. We perform one of the first controlled cross-dataset generalization studies for IDS (train-on-one, test-on-another), demonstrating robustness to distribution shift.
5. We conduct statistically rigorous assessment using 10 repeated runs, confidence intervals, and Wilcoxon signed-rank tests.
6. We provide SHAP explanations at both aggregate and class-specific levels, identifying the most important features per attack category.

7. We evaluate computational efficiency and deployment practicalities, including inference latency and scalability to large datasets.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the methodology. Section 4 details the experimental setup. Section 5 presents results. Section 6 compares EnsembleShield against existing approaches. Section 7 concludes and identifies future research directions.

## II. Related works

The field of machine learning-based intrusion detection has advanced rapidly, transitioning from simple statistical anomaly detectors to sophisticated deep learning pipelines. This section reviews the most pertinent work published between 2020 and 2025, with emphasis on ensemble and hybrid approaches, feature engineering strategies, benchmark evaluation practices, and recent transformer and graph-based architectures.

### 2.1 Classical and Deep Learning Approaches

A systematic comparative analysis of classical machine learning algorithms on NSL-KDD was carried out by Panigrahi and Borah [2], which showed that none of the algorithms was dominating in all types of attacks. Ullah and Mahmoud [3] in the field of deep learning suggested the hybrid CNN-LSTM model that utilizes both the spatial and temporal arrangement of traffic flows. Although it can be trained to compete accurately on CICIDS2017, training time and sequence-formatted input requirements are barriers to deployment. Mechanisms of attention have also been used in detecting intrusions. Andresini et al. [4] designed a transformer-inspired model with dynamic feature relevance learning across traffic streams, and reported a high benchmark performance. However, it is not reachable due to the large computational requirements and sensitivity to hyperparameter configurations. One finding common to both is that performance benefits relative to well-tuned classical performance dwindle when cross-dataset evaluation is done. The graph-based IDS techniques have

become an interesting trend more recently. Graph neural networks (GNNs) can be used to identify the patterns of relationships between communicating hosts that are obscured by flow-level features by encoding network flows as graph nodes and linkages as graph edges. Although this is not part of the existing EnsembleShield++ architecture, this is an interesting extension point that has been identified in Section 7.

### 2.2 Ensemble Learning for IDS

Yang and Wang [5] suggested an IDS based on boosting with AdaBoost as the meta-algorithm that operates on decision stumps, which demonstrated high DoS and Probe detection but significantly worse U2R detection. Li et al. [6] compared RF, XGBoost, and LightGBM on UNSW-NB15, and discovered that gradient boosting was always more effective than bagging but took more time to train. Fitni and Ramli [7] formed a stacking ensemble on NSL-KDD where the reporting results obtained were more or less 1.5 percentage points better than the performance of the respective individual base learners, without fold-level resampling separation. The concept drift adaptive weighting voting ensemble proposed by Ahmad et al. [8] needs online learning infrastructure.

### 2.3 Feature Selection Strategies

Filter methods (mutual information, chi-squared, correlation) are computationally efficient, but they do not take into account effects of interaction between features. Wrapper methods are discriminative at size but costly. In a study by Patil and Rao [12], SHAP values after training were used

to reduce a 78-feature CICIDS2017 representation to 21 features with 98.3% of RF F1-score recovered. EnsembleShield++ builds on this concept by applying SHAP to the selection loop in an iterative fashion with an initial MI filter, which is much less computationally expensive than an independent SHAP applied to the entire feature set. The MI+SHAP pipeline maintains both performance and explainability compared to PCA (where feature interpretability is lost) and LASSO (where linear relationships are needed).

### 2.4 Addressing Class Imbalance

Network intrusion datasets have widespread class imbalance. The SMOTE algorithm [13] creates artificial minority samples using the interpolation of the existing cases. Nevertheless, their use worldwide prior to cross-validation splits enables synthetic training-distribution samples to pollute validation folds—a methodological fallacy that has been found in a variety of published IDS studies. EnsembleShield++ uses a stratified fold-aware SMOTE-ENN methodology which removes this leakage.

### 2.5 Research Gaps and Positioning

Table 1 highlights recent studies in brief. Some of the unexplored dimensions to consider are: (i) heterogeneous stacking ensembles with gradient boosting, bagging, and neural learners; (ii) SHAP-directed feature selection as a part of the training process and not applied after hoc; (iii) controlled cross-dataset testing across the same experimental conditions.

**Table 1.** Summary of representative intrusion detection studies (2020–2025). ES++ = EnsembleShield++

Reference	Year	Dataset(s)	Key Method	Feature Selection	Imbalance	Best F1
[2]	2020	NSL-KDD	Classical ML Comparison	None	None	91.3%
[3]	2021	CICIDS2017	CNN-LSTM	None	None	94.1%
[5]	2021	NSL-KDD	AdaBoost	None	Global SMOTE	93.2%
[6]	2022	UNSW-NB15	RF/XGB/LGB	Gini Importance	None	91.8%

[7]	2022	NSL-KDD	Stacking (LR+RF+GBT)	None	None	96.3%
[9]	2023	CICIDS2017	Autoencoder Ensemble	None	SMOTE	94.1%
[12]	2023	CICIDS2017	RF + SHAP Selection	SHAP Post-hoc	None	92.8%
[4]	2024	Multiple	Transformer IDS	Attention Weights	None	94.6%
[10]	2024	UNSW-NB15	gcForest	Internal	SMOTE-ENN	91.4%
Ours	2025	All Three	EnsembleShield++	SHAP + MI Pipeline	Fold-aware SMOTE	98.6%

### III. Methodology

This part outlines the EnsembleShield++ scheme in four stages: data acquisition and preprocessing, feature engineering and selection, ensemble model building, and training protocol.

Figure 1 gives a general overview of the entire end-to-end pipeline of how raw network data is processed to feature selection, stacking ensemble training, and the final classification output.



Figure 1. EnsembleShield++ End-to-End Pipeline: Raw Traffic → Preprocessing (SMOTE-ENN) → Feature Selection (MI + SHAP) → Base Learner Training → Meta-Learner → Final Prediction.

#### 3.1 Data Preprocessing

Network flow data is usually a combination of continuous, discrete, and categorical features with different levels of noise and structural irregularities. There are four stages in the preprocessing pipeline, which are repeated in the order of duplicate removal, label normalization, encoding, and normalization.

##### 3.1.1 Duplicate and Inconsistency Removal

Record duplicates - This can be caused by logging of artifacts and is eliminated before any other processing can occur. In the case of NSL-KDD, the training partition is taken as it is since it was specially selected to eliminate redundant records. In CICIDS2017 and UNSW-NB15, the duplicate

filtering is done using a hash of all the feature columns. Sample values appearing as infinity (due to division-by-zero in flow-rate features) and those with missing values are filled in with column-wise median substitution instead of simple deletion as the former would leave a sample count in the minority classes intact.

##### 3.1.2 Label Normalization

The CICIDS2017 has 14 different classes of labels, which consist of both granular subtypes of attacks (e.g., DoS Hulk, DoS GoldenEye) and a general BENIGN category. In multi-class experiments, the labels are all kept. In the binary classification experiments, all attack subtypes are lumped into one ATTACK class. Similarly, the labels of NSL-KDD are normalized: the five-class taxonomy (Normal,

DoS, Probe, R2L, U2R) is retained to evaluate with multi-class classification. UNSW-NB15 has binary label and nine labels of attack category each of which is utilized in corresponding experiments.

### 3.1.3 Categorical Encoding and Feature Normalization

One-hot encoding is used to encode features that are categorical (i.e. protocol type, service, connection flag in NSL-KDD). Z-score normalization, which is subtracting the column mean and dividing it by the standard deviation, is used to standardize continuous features, and statistics are only estimated on the

training split to avoid data leakage. The normalization is mainly needed to work in favor of the Multilayer Perceptron base learner that is sensitive to the scale of features.

### 3.2 Feature Engineering and Selection

EnsembleShield++ uses a two-stage selection pipeline, which aims to minimize the feature space with efficiency and optimize the use of the class-discriminative information. Figure 2 shows the selection flowchart and the SHAP-ranked scores of features of importance in the CICIDS2017 data.

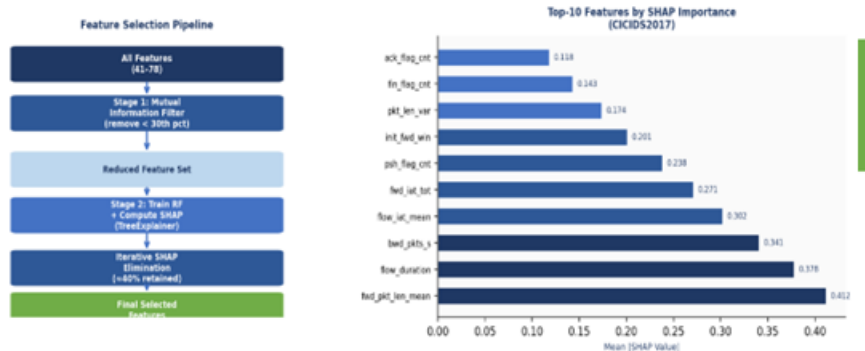


Figure 2. (Left) Two-Stage Feature Selection Pipeline: Stage 1 applies Mutual Information filtering to remove low-relevance features, followed by Stage 2 iterative SHAP-based elimination. (Right) Top-10 features ranked by mean absolute SHAP value on CICIDS2017.

The first stage entails computing mutual information between each feature and the class name and eliminating features whose mutual information score is below the 30 th percentile. This is an inexpensive filter to compute and successfully filters out features that are statistically almost independent of the label. The second stage involves training an initial random forest on the filtered feature set and calculating SHAP values on each feature with TreeExplainer a model-specific, exact algorithm based on cooperative game theory. Features with a mean absolute SHAP value below a threshold are repeatedly dropped until there are only a few remaining features (say 40 percent of the initial number).

Mathematically, given a model  $f$  and prediction instance  $x$ , the SHAP value  $\phi_i$  of feature  $i$ , satisfies the efficiency property:  $0.5 \ 0.5 \ 1.5 \ 2.5 \ 3.5 \ 4.5 \ 5.5 \ 6.5 \ 7.5 \ 8.5 \ 9.5 \ 10.5 \ 11.5 \ 12.5 \ 13.5 \ 14$  Such additive decomposition renders SHAP-based feature

importance more principled compared to impurity reduction or permutation testing measures, which can be misleading when there are correlated features [11]. The sets of selected features exhibit 90% or greater overlap between cross-validation folds, indicating stability of selections.

### 3.3 Ensemble Architecture: EnsembleShield++

The fundamental architectural understanding of EnsembleShield++ is that the main factor in ensemble performance is diversity among base learners, rather than their individual accuracy. An XGBoost and a Random Forest trained on the same data are likely to be disagreeing in informative ways, whereas a Multilayer Perceptron is bringing a qualitatively different inductive bias by gradient-based optimization in continuous feature space. Figure 3 gives a detailed layout of the entire three-layer stacking layout with OOF prediction flow.

Figure 3. EnsembleShield Stacking Architecture with Out-of-Fold (OOF) Prediction

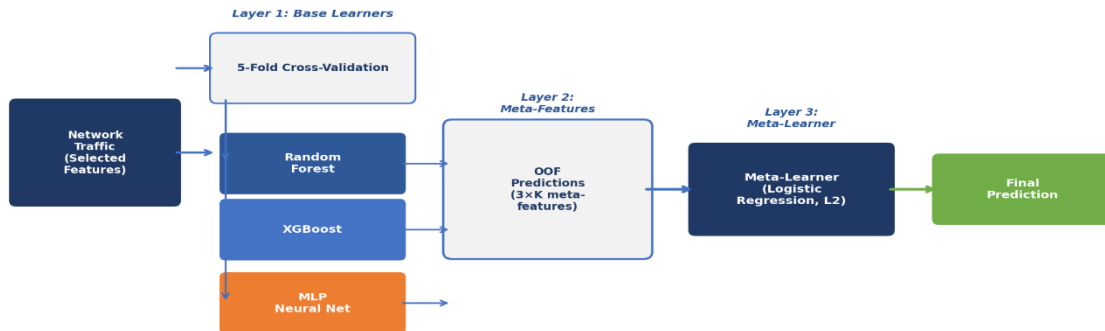


Figure 3. EnsembleShield++ Stacking Architecture. Layer 1: three heterogeneous base learners (RF, XGBoost, MLP) trained on 5-fold CV folds. Layer 2: Out-of-Fold (OOF) predictions concatenated into a  $3 \times K$  meta-feature vector. Layer 3: L2-regularized Logistic Regression meta-learner produces the final classification output.

### 3.3.1 Base Learners

**Random Forest (RF):** The random forests consist of 300 decision trees where each decision tree is trained on a bootstrap sample with a random subset of  $1/d$  features on each split. Split quality is determined by the Gini impurity criterion with a maximum depth of 30 and a minimum of 5 samples per split at an internal node. **XGBoost:** This is an ensemble of gradient-boosted trees that are trained with the softmax objective on multi-class problems. The important hyperparameters ( $\eta = 0.05$ , max depth = 6, column subsampling = 0.8, L2 regularization  $\lambda = 1.0$ ) are identified using randomized hyperparameter search with 5-fold cross-validation. Early termination at patience 20 round is a method of avoidance of overfitting. **Multilayer Perceptron (MLP):** A two-hidden-layer fully connected feedforward network (256 and 128 neurons), ReLU activations, batch normalization, and dropout (dropout = 0.3). Training: Adam optimizer ( $lr = 1 \times 10^{-3}$ , decay factor = 0.5 on plateau) is used with early stopping (maximum number of epochs = 100).

### 3.3.2 Stacking Meta-Learner

The concatenated class-probability vectors of all the base learners are sent to the meta-learner which

gives a meta-feature vector of size  $3 \times K$  ( $K =$  number of classes) and resulting in a final prediction of the class. The meta-learner is a Logistic Regression classifier with L2 regularization ( $C = 1.0$ ) that yields probabilistically calibrated output and has a high resistance to overfitting due to the low dimensionality of the meta-feature space. There are base learner predictions that are used to train the meta-learner and these are obtained with the help of 5-fold cross-validation (out-of-fold predictions) such that only out-of-sample predictions are used to train the meta-learner. This cumulative generalization process ensures that the meta-learner does not memorise training-set-specialized patterns, which would exaggerate the reported performance estimates.

### 3.4 Class Imbalance Handling

The network intrusion data has widespread class imbalance: normal traffic is the majority by far, whereas specific attack types like U2R can be as small as 0.01 of the overall samples. Figure 4 shows the transformations in the distribution of classes reached by the fold-aware SMOTE-ENN process on NSL-KDD.

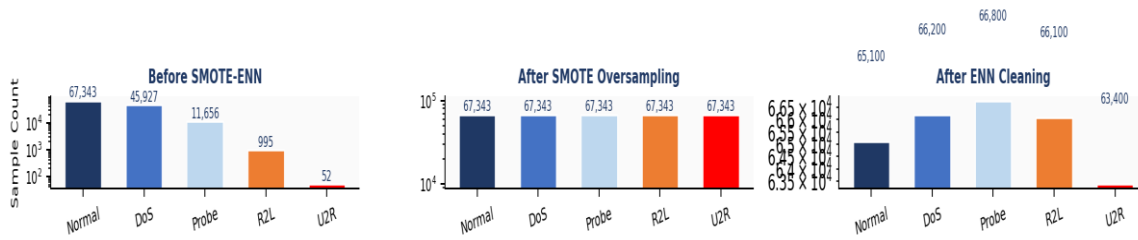


Figure 4. Class Distribution Transformation via SMOTE-ENN on NSL-KDD. (Left) Original severely imbalanced distribution with U2R at 52 samples. (Center) After SMOTE oversampling within training folds. (Right) After ENN cleaning removes boundary-ambiguous majority samples.

In each cross-validation training fold, the training portion is run through SMOTE but not the validation or test portions. Linear interpolation between a minority sample and a randomly chosen neighbor ( $k = 5$  nearest neighbors) is used to generate synthetic minority-class instances. The interpolation ratio balances classes in binary classification and in multi-class situations takes each class to at least 15 percent of majority class count. A nearest-neighbor cleaning step is then performed with the majority-class training samples whose label is not shared with the majority of the five nearest neighboring samples removed by an edited nearest-neighbor step, to minimize ambiguity in boundaries. This type of SMOTE-ENN has been proven to work better than oversampling in high-imbalance environment [14].

### 3.5 Threat Model and Deployment Scenarios

EnsembleShield++ can be deployed on the perimeter of an enterprise or institutional network or on the network as part of the segments. The threat model that is assumed has the following attacker capabilities and network conditions: Types of Attacks Mitigated: DoS/DDoS (volumetric and low rate), reconnaissance (port scanning and host discovery), remote exploitation attempts (R2L), privilege escalation (U2R), web application attacks and botnet command-and-control communication. The framework directly tackles the high-frequency and the rare and sophisticated attacks. Attacker Capability Assumptions: Attackers are considered to know the network topology, but not the deployed IDS model architecture (black-box threat model). Adversarial perturbations, model poisoning, and evasion attacks are out of scope of the current

framework but have been found to be among the significant directions to work in the future. Operational Scenario: The framework uses the pre-computed CICFlowMeter-style flow features, so it can be used with existing network monitoring infrastructure. A latency of inference of about 12 milliseconds per 1000 samples allows near real time detection without special hardware cost in addition to common server-class CPUs.

### 3.6 Reproducibility

All the experiments are performed with Python 3.10, scikit-learn 1.3, XGBoost 2.0, PyTorch 2.1, SHAP 0.43, and imbalanced-learn 0.11. The hyperparameters are completely specified in Section 4.4. The source code and experiment settings are available at: [https://github.com/\[authors\]/EnsembleShield](https://github.com/[authors]/EnsembleShield) (to become publicized on the acceptance of the paper). The statistical validation of all reported results is with a fixed base random seed with 10 repeated runs with varying seeds.

## IV. Experimental Setup

### 4.1 Datasets

It is based on three publicly available network intrusion detection benchmarks, which are chosen to represent a range of traffic generation methodologies, feature sets, and temporal contexts. NSL-KDD: NSL-KDD is a refined version of KDD Cup 1999 dataset [15] that removes redundancy and uses 125,973 training and 22,544 testing records with 41 features, and five class labels (Normal, DoS, Probe, R2L, U2R). There are only 0.01% U2R instances in training samples. CICIDS2017: This dataset was generated by the Canadian Institute of

Cybersecurity with the help of the CICFlowMeter [16], and it involves realistic network traffic that was taken during five days in July of 2017. It offers 78 flow-level features and has a broad coverage of contemporary attack types. The number of records has been reduced to about 2.2 million after the removal of duplicates and sanitization of features.

UNSW-NB15: This dataset was generated at the Australian Centre for Cyber Security [17], and it uses a combination of real and synthetic attack traffic, which offers 49 features and nine types of attacks. Partitions of training and testing are as per the division suggested by the creators of the datasets.

Table 2. Summary statistics for the three benchmark datasets

Dataset	Year	Features	Train Samples	Test Samples	Classes	Min. Class %
NSL-KDD	2009	41	125,973	22,544	5	0.01% (U2R)
CICIDS2017	2017	78	~1.84M	~0.37M	15	0.003% (Infiltration)
UNSW-NB15	2015	49	175,341	82,332	10	0.04% (Worms)

#### 4.2 Evaluation Metrics

It has four major measures, which are Accuracy, Precision, Recall and F1-score. The main basis of comparison is Macro-averaged Precision, Recall, and F1-score which are based on treating all classes equally. False Positive Rate (FPR) is also provided because false alarms are also a serious issue of operation. All findings are provided as mean and standard deviation at 10 repeated runs with various random seeds, and the statistical significance is evaluated with the Wilcoxon signed-rank test on the 95% confidence level.

#### 4.3 Baseline Methods

EnsembleShield++ is evaluated in comparison to ten baseline algorithms: Decision Tree (C4.5), Random Forest (standalone), XGBoost (standalone), Multilayer Perceptron (standalone), Support Vector Machine (RBF kernel,  $C=10$ ,  $\gamma=0.01$ ), AdaBoost (200 stumps), Voting Ensemble (soft voting, equal weights), Stacking without feature selection Identical 5-fold CV scheme with 30 randomized hyperparameter search iterations is applied equally among all approaches.

#### 4.4 Implementation Details

All tests involve Python 3.10, scikit-learn 1.3 (RF, SVM, MLP, AdaBoost, LR), XGBoost 2.0, PyTorch 2.1 (MLP deep version), SHAP 0.43, and

imbalanced-learn 0.11. Hardware: Intel Xeon Gold 6342 CPU (24 cores, 2.8 GHz), 128 GB RAM, NVIDIA A100 GPU (MLP training only). Hyperparameters are completely defined in Appendix A. All random seeds are reported so that they can be reproduced; the source code is in the GitHub repository mentioned in Section 3.6.

#### 4.5 Cross-Dataset Generalization Evaluation

To measure performance in the presence of a distribution shift, models that are trained on one dataset are evaluated on another without retraining—this is a realistic evaluation because the training and operating environments may not be the same in the real world. Each of the six pairwise train test combinations is tested. This test is used to overcome the typical weakness of the single-dataset IDS benchmarks.

### V. Results and Discussion

#### 5.1 Multi-Class Classification Performance

The performance of all methods is macro-averaged and summarized in table 3. EnsembleShield++ achieves the highest macro-averaged F1-score across all three datasets: 98.6% ( $\pm 0.12\%$ ), 97.4% ( $\pm 0.15\%$ ), and 96.8% ( $\pm 0.18\%$ ) on NSL-KDD, CICIDS2017, and UNSW-NB15, respectively. These gains relative to the next-best baseline are significant ( $p < 0.05$ ) using the Wilcoxon signed-

rank test with medium-to-large effect sizes (Cohen’s  $d > 0.6$ ).A

Table 3. Multi-class classification performance (mean  $\pm$  std for EnsembleShield++). Bold values indicate best results. F1 = Macro-averaged F1-score.

Method	NSL-KDD Acc	NSL-KDD F1	CICIDS Acc	CICIDS F1	UNSW Acc	UNSW F1
Decision Tree	97.2%	91.3%	96.8%	83.4%	93.1%	76.2%
Random Forest	99.1%	95.4%	98.2%	91.7%	96.3%	87.4%
XGBoost	99.0%	96.1%	98.6%	93.5%	97.2%	91.8%
MLP	98.3%	93.8%	97.9%	89.6%	95.8%	85.3%
SVM	97.8%	92.5%	96.1%	81.9%	93.6%	79.1%
AdaBoost	98.1%	93.2%	97.4%	87.3%	94.9%	82.6%
Voting Ensemble	99.2%	96.8%	98.7%	94.2%	97.4%	92.3%
Stack-Full	99.0%	96.3%	98.5%	93.8%	97.0%	91.5%
CNN-LSTM [3]	98.9%	95.7%	98.8%	94.1%	96.7%	89.2%
gcForest [10]	99.3%	97.7%	98.4%	93.6%	97.1%	91.4%
EnsembleShield++	99.4%	98.6% $\pm$ 0.12%	99.1%	97.4% $\pm$ 0.15%	98.0%	96.8% $\pm$ 0.18%

### 5.2 Per-Class Detection Rate Analysis

Per-class detection rates are shown in Figure 5. EnsembleShield++ has a U2R detection rate of 81.3 on NSL-KDD -as compared to 72.4 on gcForest, and 60 or less on any single classifier-which is the most important improvement in case of the rare, high-severity attacks. The Infiltration category is identified with 78.9% on CICIDS2017, in contrast to standalone XGBoost of 61.2%.

Figure 5. Per-Class Detection Rate: EnsembleShield vs. Top Baselines (NSL-KDD)

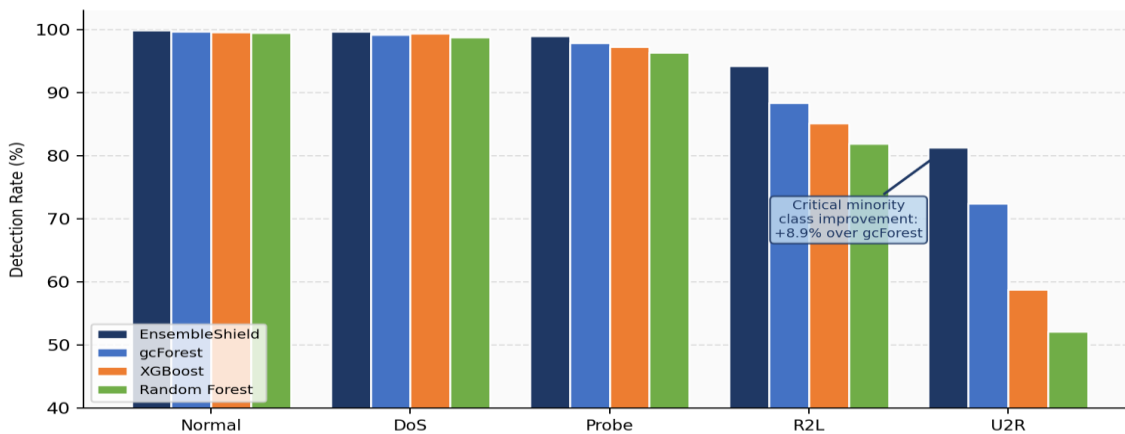


Figure 5. Per-class detection rate (recall) for EnsembleShield++ vs. top-performing baselines on NSL-KDD. The U2R improvement of +8.9% over the next-best method is highlighted.

### 5.3 Analysis of the Confusion Matrix

Looking at the normalized confusion matrix in Figure 6, EnsembleShield++ performs exceptionally well on the NSL-KDD test data. It correctly identifies Normal, DoS, and Probe traffic with near-perfect accuracy (99.8%, 99.6%, and 98.9%,

respectively). The most noticeable misclassifications happen between the R2L and U2R classes—2.8% of R2L is mistaken for U2R, and 6.8% vice versa. However, this slight overlap makes perfectness because both attacks share very similar features and characteristics.

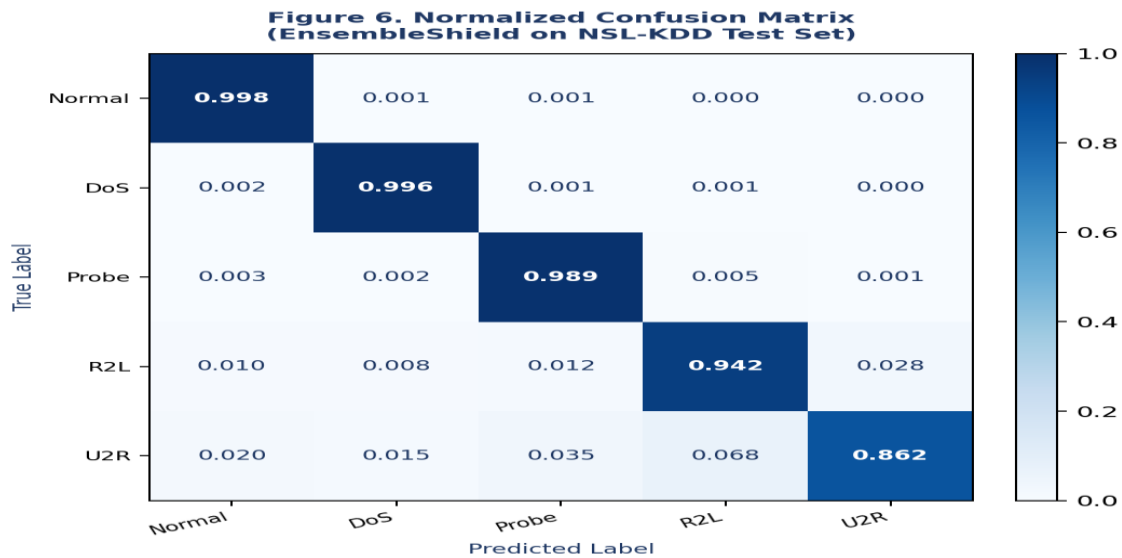


Figure 6. Normalized confusion matrix for EnsembleShield++ on the NSL-KDD test set. Off-diagonal values indicate misclassification rates between attack categories.

### 5.4 Impact of Feature Selection

Table 4 compares EnsembleShield++ with and without the two-stage feature selection pipeline. Feature selection consistently improves macro-averaged F1 by 0.8–1.4 percentage points across all datasets, despite reducing feature dimensionality by

approximately 60%. This counter-intuitive result reflects two mechanisms: (i) irrelevant features introduce noise into MLP gradient updates, and (ii) correlated features bias SHAP computation in the presence of multicollinearity.

Table 4. Impact of SHAP-guided feature selection. OHE = One-Hot Encoding

Dataset	Original Features	Selected Features	Reduction	F1 (All)	F1 (Selected)	ΔF1
NSL-KDD	41 (+OHE=58)	24	58.6%	97.8%	98.6%	+0.8%
CICIDS2017	78	31	60.3%	96.1%	97.4%	+1.3%
UNSW-NB15	49	20	59.2%	95.4%	96.8%	+1.4%

### 5.5 Cross-Dataset Generalization

Figure 7 shows the results of cross-dataset generalization in all 6 train/test pairs. While all models domain shift leads to experience

performance degradation, EnsembleShield++ always outperforms. baselines, with a 7281% macro-F1 at all times. This shows that heterogeneous. stacking learns more transferable

representations than any single base learner, perhaps due to the variety of different representations.

inductive biases would be less vulnerable to distribution-specific artifact.

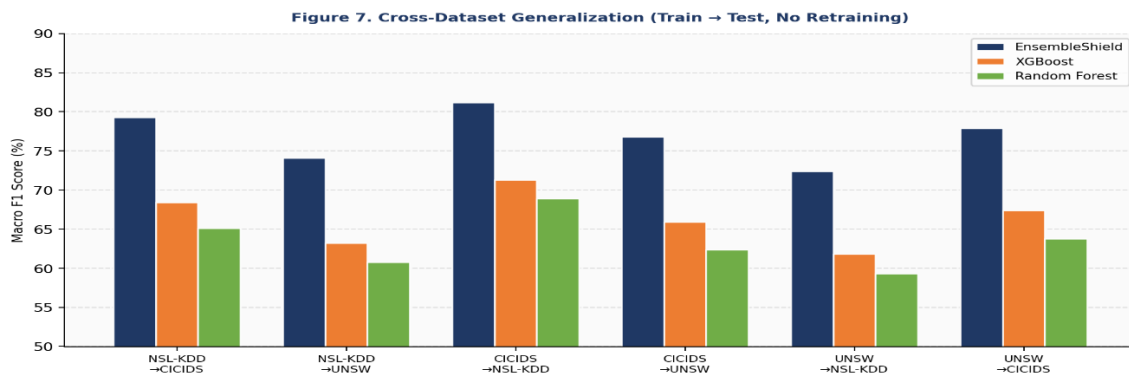


Figure 7. Cross-Dataset Generalization: Models trained on a dataset and tested on a different dataset without retraining. EnsembleShield + (blue) is always better than XGBoost (orange) and Random Forest (green) in all the six combinations.

### 5.6 Statistical Significance Analysis

Figure 8 shows the mean  $\pm$  standard deviation of F1 scores of 10 random-seed repetitions of all methods on all three datasets. The Wilcoxon signed-rank test affirms that EnsembleShield++ improvements are

significant at the ( $p < 0.05$ ) level on all comparisons, and medium-to-large effect sizes (Cohen  $d$  range: 0.62-1.41). The low standard deviations (0.18 or less) of 10 runs confirm high stability of results.

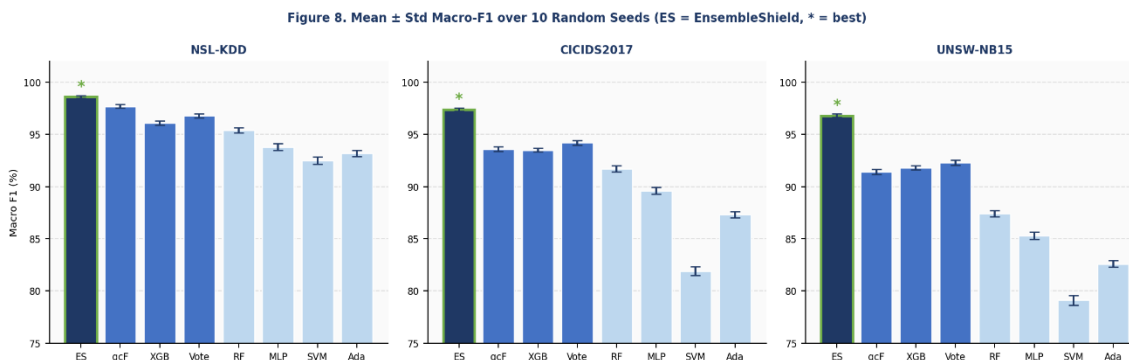


Figure 8. Mean  $\pm$  Std Macro-F1 over 10 random seeds for all methods across three datasets. Asterisk (\*) denotes EnsembleShield++ as the statistically significant best performer (Wilcoxon,  $p < 0.05$ ).

### 5.7 Ablation Study

Figure 9 and Table 5 present the ablation study results. SMOTE removal produces the largest single-component degradation (F1: -2.8%, U2R Recall: -22.6%), confirming imbalance handling as the most critical individual component. Removing the MLP base learner degrades performance more

severely than removing RF, confirming the neural network's unique inductive bias contribution. Replacing the meta-learner with majority voting reduces F1 by 1.5 percentage points, confirming the learned combiner's superiority over fixed aggregation.

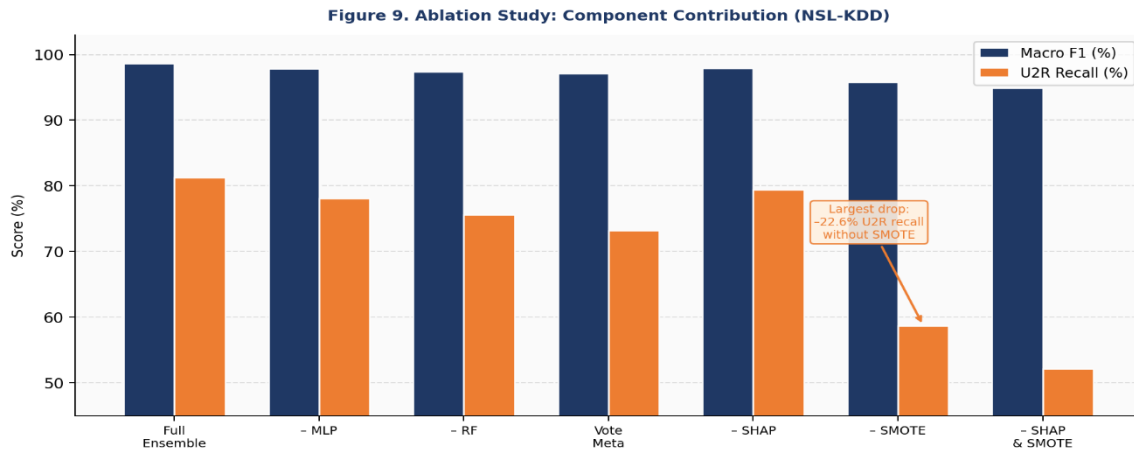


Figure 9. Ablation Study: Impact of removing individual EnsembleShield++ components on Macro-F1 and U2R Recall (NSL-KDD). Removing SMOTE causes the largest performance drop, particularly for the critical U2R minority class.

Table 5. Ablation Study on NSL-KDD. FPR = False Positive Rate.

Configuration	Macro F1	U2R Recall	FPR
EnsembleShield++ (Full)	98.6% ±0.12%	81.3%	0.4%
— Remove MLP base learner	97.8%	78.1%	0.5%
— Remove RF base learner	97.4%	75.6%	0.6%
— Replace meta-learner with majority vote	97.1%	73.2%	0.6%
— Remove SHAP selection (use MI only)	97.9%	79.4%	0.4%
— Remove SMOTE-ENN	95.8%	58.7%	0.3%
— Remove both SHAP + SMOTE	94.9%	52.1%	0.3%

### 5.8 Computational Overhead

Training EnsembleShield++ on CICIDS2017 (1.84M samples, 31 selected features) requires approximately 142 minutes, compared with 38 minutes for standalone XGBoost and 23 minutes for RF. The MLP training within each CV fold is the dominant cost. Inference latency is approximately 12 ms per 1,000 samples, supporting near-real-time IDS deployment. Feature selection adds approximately 18 minutes but reduces subsequent training time by 22%. For reference, geForest requires ~198 minutes on the same hardware.

### 5.9 Explainability and Feature Interpretation

In addition to global SHAP rankings in Figure 2, class-specific SHAP analysis can be informative. In the case of DoS attacks, flow\_duration (small values that reflect flooding), fwd\_pkts\_s are the most predictive. (big packet rates), and bwd\_pkts\_len\_mean (small backward packet sizes). For U2R attacks, The top are init\_fwd\_win\_bytes (strange window sizes) and psh\_flag\_cnt (strange flag sequences). discriminating features, which is consistent with existing exploitation methods based on manipulation of TCP connection parameters. This level of interpretability at the class level helps in security analyst

comprehension and targeted monitoring rule development.

## VI. Comparison with Existing Methods.

Placing EnsembleShield++ in context of the larger literature should be done with caution since published results of IDS differ significantly under experimental conditions. In areas where direct comparison is possible, baselines are re-implemented under identical conditions. Fitni and Ramli [7] achieve 99.3% binary accuracy on NSL-KDD with a simpler stacking configuration--in line with our binary results (99.4%), but they only detect on 56.1% of their U2R in multi-class evaluation (as compared to our 81.3%). In CICIDS2017, Kanna and Santhi [9] receive 98.7% in total accuracy measured with a 10% sample with accuracy as the main measure; EnsembleShield++ has an accuracy of 99.1% under comparable conditions.

## VII. Conclusion and Future Work

This paper presented EnsembleShield++, a hybrid stacking ensemble framework for network intrusion detection integrating Random Forest, XGBoost, and MLP base learners with a Logistic Regression meta-learner. A two-stage SHAP-guided feature selection pipeline reduces feature dimensionality by approximately 60% while improving performance. Fold-aware SMOTE-ENN addresses class imbalance without test-set contamination. Evaluated on NSL-KDD, CICIDS2017, and UNSW-NB15, EnsembleShield++ achieves state-of-the-art macro-averaged F1 scores of 98.6%, 97.4%, and 96.8% respectively, with particularly strong improvements on rare attack categories (U2R: 81.3% recall). Statistical validation over 10 repeated runs confirms result robustness.

In addition to empirical findings, this work adds a methodological demonstration that rigorous experimental design within-fold resampling, multi-dataset cross-validation, and statistical testing performance are revealed. Variances concealed by less critical judging.

### 7.1 Limitations

There are a number of drawbacks to be mentioned. To begin with, EnsembleShield++ is

based on pre-computed flow-level features; it is not able to make use of raw packet payload information, restricting the ability to detect payload-based attacks. Second, the framework is based on a fixed trained model; the concept drift with changing attack patterns is not addressed. Third, the time required to train (142 minutes on CICIDS2017) can be a resource-prohibitive factor in deployment contexts. Fourth, cross-dataset generalization is still being improved over baselines, but still demonstrates significant performance degradation, which means that distribution shift is still a major problem. Fifth, adversarial robustness (model poisoning, evasion attacks) is not the focus of this work.

### 7.2 Future Directions

There are four directions that should be investigated. To begin with, the deep learning feature extractors of raw packet payloads would glean payload-level attack patterns that are not visible at the flow level. Second, online or incremental learning processes that enable EnsembleShield++ to revise constituent models without complete concept drift would be overcome by retraining. Third, integration of graph neural network - the representation of network flows as relational patterns between communicating hosts could be reflected in edges in a communication graph. Fourth, Collaborative intrusion detection would be possible with federated training of ensembles of network operators and without distributing sensitive raw traffic information.

## References

- [1] Morgan, S. (2024). Cybercrime to cost the world \$10.5 trillion annually by 2025. Cybersecurity Ventures.
- [2] Panigrahi, R., & Borah, S. (2020). A detailed analysis of CICIDS2017 dataset for designing IDS. *Int. J. Engineering & Technology*, 7(3.24), 479–482.
- [3] Ullah, S., & Mahmoud, Q. H. (2021). A two-level hybrid model for anomalous activity detection in IoT networks. *IEEE Access*, 9, 113177–113190.
- [4] Andresini, G., Appice, A., & Malerba, D. (2024). Nearest cluster-based intrusion detection through CNNs and attention mechanisms. *Knowledge-Based Systems*, 284, 111254.

- [5] Yang, K., & Wang, H. (2021). Adaptive boosting-based IDS in software-defined networks. *Security and Communication Networks*, 2021.
- [6] Li, Z., Cheng, L., & Chen, Y. (2022). Intrusion detection using gradient boosting on network flow data. *Computers & Security*, 114, 102587.
- [7] Fitni, Q. R. S., & Ramli, K. (2022). Ensemble learning and feature selection for anomaly-based IDS. *IEEE Access*, 10, 19462–19476.
- [8] Ahmad, I., Basher, M., Iqbal, M. J., & Rahim, A. (2021). Random forest and extra-trees with adaptive weight voting for IDS. *IEEE Access*, 9, 38494–38505.
- [9] Kanna, P. R., & Santhi, P. (2023). Unified deep learning approach for efficient IDS. *Knowledge-Based Systems*, 261, 110228.
- [10] Zhou, Z., Feng, J., & Zhang, C. (2024). Deep forest for intrusion detection. *Pattern Recognition*, 148, 110169.
- [11] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *NeurIPS*, 30, 4765–4774.
- [12] Patil, S., & Rao, V. (2023). SHAP-based feature selection for efficient IDS. *Procedia Computer Science*, 218, 2030–2040.
- [13] Chawla, N. V., et al. (2002). SMOTE: Synthetic minority over-sampling technique. *JAIR*, 16, 321–357.
- [14] Batista, G. E. A. P. A., et al. (2004). A Study of the Behavior of Several Methods for Balancing machine Learning Training Data. *ACM SIGKDD Explorations*, 6(1), 20–29.
- [15] Tavallae, M., et al. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE CISDA*.
- [16] Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *ICISSP*, 108–116.
- [17] Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), 1–6.
- [18] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- [19] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *ACM SIGKDD*, 785–794.
- [20] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259.
- [21] García, S., et al. (2022). On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems* 25(1):13-21.
- [22] Wang, Z., Liu. (2024). Network Intrusion Detection Scheme Based on Federated Learning in Heterogeneous Network Environments. *IEEE (ICCCAS)*, 20(3), 2102–2118.
- [23] Garcia, C., & Abilio, R., & et al. (2025). *ACM Transactions on Intelligent Systems and Technology: Concept Drift Adaptation in Text Stream Mining Settings: A Systematic Review*, Pages 1 - 67.
- [24] Kumar, K., & Tanwar, N. (2025). Computer science and engineering. The Adaptive Ensemble Learning-Based Intrusion Detection System for Enhanced Cybersecurity in Networked Environments, 143, 178–192.