Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.086208

ARTICLE

Context-Aware Ensemble Learning with Adaptive Attention for High-Fidelity Network Intrusion Detection

Venkata Mohit Tamanampudi[1], **Zaid Bin Faheem**[2] **and Jehad Ali**[3,*]

[1]JPMorgan Chase, New York, NY, USA
[2]Department of Computer Science, Wuhan University, Wuhan, China
[3]Department of AI Convergence Network, Ajou University, Suwon, Republic of Korea

*Corresponding Author: Jehad Ali. Email: jehadali@ajou.ac.kr

Received: 26 May 2026; Accepted: 03 August 2026; Published: 22 October 2026

ABSTRACT: Intrusion Detection Systems (IDS) are an important tool for network security, however, machine learning based IDS and single-based models often fail to be beneficial in the context of high-dimensional data, redundant features, class distributions, and flexibility when network attack patterns change. To address these drawbacks, the present paper proposes a novel ensemble learning framework called CAF-Net (Context-Aware Fusion Network), which combines bagging, stacking, an attention-driven meta-learning and a confidence-aware soft voting mechanism based on the XGBoost ensemble learning approach. Although CAF-Net uses the same base learners as traditional ensemble methods, it differs in that it assigns importance weights to the predictions of each learner that depend on the instances, and in that other factors besides the context are taken into account when fusing the decisions, such as the trustworthiness of the model. The framework uses the mutual information-based feature selection method to distinguish the characteristics of the network with the discrimination of the attributes that are not essential to the network behavior. Specifically, from the 42 original features present in the NSL-KDD dataset, mutual information scores are computed for each feature, and the top 15 features with the highest discriminative power are selected for model training. In addition, CAF-Net implements a four-stage ensemble learning framework: Stage 1 uses bagging with XGBoost to decrease prediction variance, Stage 2 applies heterogeneous stacking with KNN and LR to capture complementary local and global decision boundaries, Stage 3 performs attention-based meta-learning to compute instance-specific importance weights, and Stage 4 performs confidence-aware soft voting to fuse decisions using attention relevance and model trustworthiness. Thorough experiments of benchmark datasets like NSL-KDD and CIC-IDS2018 covering both legacy and modern networks. CAF-Net attains a state-of-the-art accuracy, precision, recall and F1-score of 99.47%, 99.47%, 99.48% and 99.47%, respectively, on NSL-KDD. The proposed framework results in high generalization on CIC-IDS2018 dataset, and with a baseline version and an ablation version, it performs better in the difficult scenario of the imbalance of classes. Ablation experiments also support that all attention-based weighting, feature selection and soft voting boost the performance. The proposed CAF-Net paradigm can successfully model complex network traffic and it is highly accurate, stable and interpretable. The scalability, desirable inference latency of 0.85 ms per sample, linear growth with the size of data sets and the small size memory requirement of 38.6 MB indicate that CAF-Net is a scalable solution with potential application in real-world intrusion detection system. Encouraging benchmark results, but further testing and validation in operational environments is important for future work.

KEYWORDS: Machine learning model; intrusion detection systems; feature selection; ensemble learning; bagging with XGBoost; model stacking

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1 Introduction

As the internet increasingly pervades every sphere of our life, network architectures have become susceptible to a variety of cyber-attacks. Thus, the detection and mitigation of such risks have now become the center of attention. For the detection and prevention of network intrusions, network intrusion detection systems, or NIDS, are employed. The exponential growth in the use of the Internet has brought an astonishing increase in attention being paid to network security concerns. One of the most important functions done by IDSs relates to anomaly detection in network data. Resilient network security requires proper understanding of attack patterns and the creation of appropriate strong security solutions. In the last years, artificial intelligence and data science are fast developing and have become effective tools in tackling some tough problems in security. Numerous AI-based network detection strategies have been published in the literature and have shown to be beneficial in improving network security. These tactics are equipped with sophisticated data analysis tools. The traditional machine learning methods would depend on inputs and come under shallow learning. Input data preprocessing would yield more accurate results. Bhavsar and Waghmare [1] illustrates that an IDS totally depends on enhanced data mining techniques to perform the correct classification and detection of intrusions. To identify network anomalies and enable the network administrator to take appropriate security steps to prevent infiltration, a number of machine learning approaches are used. Rastogi et al. [2] and Repalle and Kolluru [3] emphasize that there is a need to adopt the application of supervised learning techniques which would classify normal from abnormal network activities. Further, such algorithms reduce false alarms in order to improve the general security of computer systems. Additionally, unsupervised learning methods are being explored in the context of identifying zero-day or unknown attacks: those that utilize previously unexploited vulnerabilities in technology. These methods examine network traffic patterns for anomalies without any prior knowledge of specific attack signatures, as they do not rely on labelled input data [4]. As cyberattacks become increasingly complex and increasingly frequent, the network security is relying on this proactive approach more and more. Introduction of feature selection methods has helped in the progress of research on IDS. In order to enhance the rate and accuracy of the intrusion detection models, dataset size is minimized and this is achieved by employing techniques used to identify the most important features within the large datasets [5]. This will make sure the IDS determines the most critical areas in regards to network traffic intrusion, hence, it will ease the process of detection streamlining as per the plausible probability of real time detection. To wrap up, the potentials of various AI-powered IDS solutions, including supervised, unsupervised and ensemble learning approaches, will usher in a new era of network security. These systems will be even more crucial to network infrastructures integrity and safety as cyber-attacks are becoming complex. Optimization of machine learning models, new feature selection techniques and fully automated anomaly detection in real-time will impact intrusion detection and network defense. In this research we have the following specific contributions:

- Comprehensive Utilization of NSL-KDD Dataset: We utilized the NSL-KDD dataset, which contains 42 original attributes covering network protocol, service type, flag indicators, and various connection-level metrics (e.g., `num_failed_logins`, `root_shell`, `num_compromised`). From these 42 features, we apply mutual information (MI)-based feature selection to objectively identify and retain the top 15 most discriminative features. It is important to note that while features such as `num_failed_logins` are present in the original dataset, our MI-based ranking assigns them relatively low discriminative power for the binary classification task; consequently, they are not retained in the final selected feature subset. The top-ranked features selected by MI are primarily byte-count, service-type, and rate-based indicators, which our analysis confirms to be the most informative for this intrusion detection task.
- Context-Aware Fusion Network (CAF-Net): We propose CAF-Net, a four-stage ensemble learning framework that extends conventional ensemble methods through complementary mechanisms. Stage 1

applies bagging with XGBoost for variance reduction. Stage 2 performs heterogeneous stacking using KNN and Logistic Regression to capture both local and global decision boundaries. Stage 3 introduces an attention-driven meta-learning component that dynamically computes instance-specific importance weights for base learner predictions, enabling context-aware adaptation to heterogeneous network traffic patterns. Stage 4 employs confidence-aware soft voting that combines attention-derived relevance scores with model trustworthiness estimates to achieve robust prediction fusion. These components collectively enhance robustness, interpretability, and generalization performance beyond traditional static ensemble aggregation approaches.

- Efficiency and Scalability Analysis: We thoroughly evaluate the computational cost of CAF-Net alongside its detection accuracy. The framework achieves an inference latency of 0.85 ms per sample (approximately 1176 samples per second throughput) on standard cloud infrastructure, with a memory footprint of 38.6 MB on disk and 245 MB in RAM, making it suitable for both high-throughput data centers and resource-constrained edge deployments. The training time scales linearly with dataset size ($R^2 = 0.994$), with an overall time complexity of $O(n \log n)$ for training and $O(n)$ for inference, predominantly due to the KNN distance computations. These quantitative results, detailed in Section 4.4, demonstrate that CAF-Net achieves real-time intrusion detection capability without sacrificing accuracy, thereby confirming its applicability to large-scale enterprise environments with high traffic volumes. This research attempts to establish a standard for future breakthroughs in intrusion detection systems, guaranteeing continuous resilience against growing cyber threats in modern network infrastructures, through establishing a link between advances in theory and real-world implementation.

While there has been some study of using bagging, stacking, attention mechanisms, and voting strategies in intrusion detection, the coordinated use of these in a single adaptive decision-making framework is not well researched. CAF-Net is different from conventional ensemble learning methods in three aspects: The first is an introduction of an attention-based meta-learner which dynamically assigns the instance-specific weights on the base learner predictions rather than specific assigning; and secondly, it proposes an attention-based instance-level self-balancing strategy that dynamically regulates the weights of instances irrespective of the class label. Second, it uses a confidence-aware soft voting procedure that incorporates contextual relevance information extracted by using the attention scores along with the model trustworthiness estimation based on the model's performance in the validations. Third, CAF-Net is proposed as a four-stage ensemble learning framework in which Stage 1 uses bagging with XGBoost to reduce variance, Stage 2 uses heterogeneous stacking with KNN and LR to minimize bias, Stage 3 performs attention-based meta-learning for instance-specific weighting, and Stage 4 performs confidence-aware soft voting as the final aggregation mechanism. This interweaving of resources allows for adaptive and interpretable intrusion detection decisions made in all kinds of network traffic.

This paper is organized as follows. Related studies are reviewed comprehensively in Section 2. Methods of data preprocessing for the intrusion detection system, evaluation metrics, a general overview of the NSL-KDD dataset, and detailed explanations of the proposed model including architecture, algorithm, and experimental results, are described in Sections 3 and 4. It also compares with previously published works. Finally, Section 5 provides the conclusion of the paper and future research directions and extensions of this research.

2 Related Work

The security of personal computers and the network are very complex challenges, and network protection is at a critical point. Because to internet browser security and protection flaws, all an attacker needs to do to take down a website is install malicious code, and users who visit these websites run a significant risk of becoming victims of these attacks [6]. Liu et al. [7] aims to bridge the gap between

traditional network security benchmarks and IoT-specific security challenges. Eleven different machine learning algorithms were tested on intrusion detection performance metrics including accuracy, MCC, and AUC. It has been observed that the best algorithms are the tree-based methods and ensemble algorithms among which XGBoost achieved an accuracy of more than 96%. The authors conclude that due to simplicity and explainability, Decision Trees seem to be a good choice for IoT devices with resource constraints. Then, Yin et al. [8] came up with an intrusion detection system that was based on an RNN, that significantly outperforms traditional methods such as J48, ANN, and SVM, demonstrating the ability of deep learning methods to raise IDS detection and accuracy rates. Chou and Jiang [9] presents a taxonomy of the main challenges and the solution for each of this challenge on network intrusion detection systems. It discusses methods for mitigating Feature-based challenges, label-based challenges, and instance-based challenges, including the feature normalization approach, incremental learning, and parallel processing techniques. Buczak and Guven [10] perform a review of techniques used for data mining and various machine learning classification used in the area of cyber intrusion detection. The paper reveals a large gap that exists in labelled data and indicates that investing in the collection and labelling of data is one path forward for ML and DM methods. In the last instance, it points to the requirement of rapid incremental learning methods to meet the high-frequency retraining occurring in cyber security applications. Ma et al. [11] proposed an algorithm which uses SC with DNN for intrusion detection in sensor networks. Their SCDNN model outperformed the traditional methods, like SVM, BPNN, and Random Forest, for six datasets with improved detection accuracy, especially those attack types that are sparse in complex networks. Raj et al. [12] proposed an IDS by utilizing CATBoost, which overcame some limitations of ELM used in the original method. A hybrid algorithm for the detection of network anomalies was discussed by Al-Safi et al. [13], presenting the combination of Information Gain-based feature selection with SVM optimized by an Artificial Bee Colony and Optimization-Cuckoo Search algorithms. The hybrid method proposed will enhance the efficiency of IDS in effective data classification and optimization of hyper-parameters of SVM. Zakariah et al. [14] uses Artificial Neural Networks to propose a new Intrusion Detection System by utilizing the NSL-KDD dataset for performance evaluation. Ahmad et al. [15] provides a detailed review on Machine learning and Deep learning methods for network intrusion detection, focusing on the recent trend of using more often the DL techniques such as the Auto-encoders and DNNs methods for their higher accuracy. Nevertheless, they have pointed out the need for modern datasets and simpler models regarding real-time applications. A deep and detailed analysis of eight benchmark datasets which include KDD99, NSL-KDD, and CSE-CIC-IDS2018 was used by Ghurab et al. [16] for evaluating NIDS. Instances, features, and classes of the datasets are elaborated, where effective use of such datasets has been provided by the authors with valuable insights and recommendations for the NIDS research. Patil and Pattewar [17] benchmark different machine learning algorithms for the KDDCup'99 and NSL-KDD datasets using AdaBoost for detection of network intrusion. Their work demonstrates that AdaBoost detects 90.00% of attacks on the NSL-KDD dataset compared to 86.27% on KDDCup'99. This also underlines the fact that NSL-KDD can be a more dependable benchmark dataset for IDS. Al-mamory and Jassim [18] uses the KDD CUP 99 dataset to present a comparison of different data mining models for pinpointing network intrusion. The performance evaluation and results indicate that Random Forest Classifier has outperformed the rest in high detection of DoS attacks, while the highest rate in the detection of Probe attacks was achieved by Fuzzy Logic. The study infers that no single algorithm best performs in all attack scenarios; hence, a blend of all is necessary for that purpose. Hu et al. in [19] developed an enhanced intrusion detection system using CNN along with an optimization algorithm called the Fruit Fly to treat class imbalance and improve recognition rates. Their approach improves the performance and accuracy when evaluated on the NSL-KDD dataset compared with a traditional CNN. In another recent study, the authors Xu et al. [20], presented a novel 5-layer auto-encoder for network anomaly detection with the model achieving highest accuracy and F1-score on the NSL-KDD dataset. Their data pre-processing provides innovation and the exclusion of available outlier's form part of their approach.

This approach provides a much better performance in detection than previous models. Almutairi et al. [21] compared various machine learning algorithms, including J48, Naïve Bayes, SVM, and Random Forest, on the NSL-KDD dataset. The study confirmed that Random Forest demonstrated superior performance in detecting network intrusions compared to the other evaluated classifiers. The authors further emphasized the accuracy and robustness of the Random Forest algorithm in both binary (normal vs. attack) and multi-class (normal vs. specific attack types) classification settings, which are the standard problem formulations for the NSL-KDD dataset.

Gurung et al. [22] investigated deep learning techniques for real-time anomaly detection to attain high detection and accuracy rates with a minimum false detection rate to ensure a robust and efficient security measure against various attack vectors. For Software Defined Networks, Alzahrani and Alenazi [23] used XGBoost, Random Forest and Decision Tree. They have applied those methodologies after performing in-depth preprocessing to maintain the quality of the data on the NSL-KDD dataset and provide a network intrusion detection system. The work is intended to refine the detection of attacks by classifying types such as DoS, PROBE, R2L, and U2R, using SDN in a flexible yet controlled manner. A deep learning model was proposed by Fu et al. [24], integrating Bi-LSTM, CNN and attention mechanisms for network intrusion detection. Their approach, experimented on the NSL-KDD dataset, established an accuracy of 90.73%, with an F1 score of 89.65%, outperforming all the other methods, and managed to show its capability to deal with the data imbalance problem. Jo et al. [25] introduced three preprocessing methods as a proposal for CNN-based IDS: direct, weighted, and compressed. Their study has resolved a couple of issues in security for IoT and ICS networks because of their heterogeneity. The methods are formulated to enhance data processing efficiency and improve network security by using advanced preprocessing techniques. Network intrusion detection solutions are designed to meet the needs of particular work contexts, such as Software Defined Networks, and therefore require adapted approaches. Magdy et al. [26] investigate the vulnerability of network intrusion detection systems by implementing the well-known NSL-KDD dataset and observe scalability and accuracy as two significant obstacles toward IoT security. They note in their comparative analysis that IDS accuracy and performance increase when supervised and unsupervised learning algorithms are combined to select key features. Javaid et al. [27] discussed the application of STL for developing a flexible and efficient NIDS, making comparisons in terms of performance metrics, which includes accuracy, precision, recall, and f1-measure against conventional methods. Patel and Sondhi [28] worked on the review of intrusion detection using machine learning algorithms coupled with the feature optimization method. The work undertook the investigation on different methods explored on various classification algorithms, neural networks, genetic algorithms, and particle swarm optimization to enhance the detection accuracy and reduce false alarms. Staudemeyer and Omlin [29] explore the performance of LSTM-RNN in classifying intrusion detection data for which they shall be able to use their time series modeling capability. Results indicate that the LSTM networks are effective for intrusion detection, with accuracy >90% particularly useful for the evaluation of network performance. Bansal and Mahapatra [30] compare three machine learning techniques for botnet detection, with a focus on the effectiveness of the DoS and other cyber-attacks. Their study tested these methods on the ISCX Intrusion Detection and CTU-13 datasets, noting that the rapid growth rate in IoT devices now substantially amplifies today's botnet threat with the severe consequences of only a few large-scale attacks, such as the 2016 Dyn DoS. Mohammed and Gbashi [31] also propose an IDS that applies RFE and combines DNN with RNN to do their magic on the famous NSL-KDD dataset. Indeed, their approach was able to perform well in binary classification as well as multi-class intrusion-detection-classification problems, by using improved feature selection with this deep learning technique. Alavizadeh et al. [32] proposed a network intrusion detection system (NIDS) leveraging a reinforcement learning approach based on Deep Q-Learning. This technique combined deep feedforward neural networks with Q-learning to improve its capability of autonomous learning. In the study by Sindhu et al. [33], the

authors' objective is to develop a lightweight IDS in detecting anomalies within the network. The key focus of the paper rests on carrying out effective preprocessing, selection of features, and designing the learning algorithm. In the IDS, there are three major objectives: to eliminate 75% of redundant data in order to avoid biased classifications, to select 16 important features using a wrapper-based algorithm, and to use neuro-tree for better classification. It includes a neural ensemble decision tree, and its performance is measured by six different decision tree classifiers that indicate an incredibly high classification rate of 98.4% with specific attack types. It generally involves significant reduction of redundant instances, optimal features selected, and robustness in the detection performance. Su et al. [34] introduce a novel BAT model, which combines BLSTM and attention mechanisms into an automatic feature learning approach for NIDS. The proposed approach based purely on deep learning outperforms other traditional approaches involving manual feature engineering. The discussed research works emphasize novelty in model integration and architecture design for enhancing intrusion detection system effectiveness. Patel et al. [35] address the challenges related to traditional IDPS mechanisms' deployment within cloud computing because of its distinct and transparent nature. This systematic review provides recent developments about IDPS and alarm management techniques that apply to cloud systems. It contributes to a comprehensive taxonomy of IDPS solutions while identifying the main requirements toward effective cloud-based intrusion detection. It proposes the use of ontology, risk management, autonomic computing, and fuzzy logic to meet these demands for improvement in cloud IDPS performance. Main highlights are: updated review regarding cloud IDPS solutions, detailed taxonomy, advanced concept applications to be used for cloud security. Tang et al. [36] proposed integration of multiple classifiers along with a Deep Stacking Network (DSN) to perform improved detection with the overcoming of the limitations of traditional methods.

The studies discussed above have concentrated on network level intrusion detection using machine learning and deep learning; an area of research related to this has considered security and resilience in networked control systems (NCS) where attacks can impact on both the communication and physical processes. Zhao and Lun [37] studied composite anti-disturbance control for networks with external disturbances and actuator attacks using event-triggered output feedback, which demonstrates how to simultaneously enhance the robustness and communication-efficiency for networked systems in adversarial settings. There are some related works on the issues of denial-of-service (DoS) attack scenarios in networked systems. Some of the existing literature focuses on resilient event triggered control for networked control systems with DoS attacks, such as Sun et al. [38] who proposed a security-based resilient event triggered control framework for networked control systems under DoS attacks, and Zhao et al. [39] who developed an adaptive event triggered control strategy with observer-based design to mitigate the influence of DoS attacks while reducing the bandwidth requirement. While these studies focus on making systems resilient using control theory, rather than packet level intrusion detection classification, they are very pertinent in pointing towards the broader requirement for making networked systems adaptive, secure and resource efficient in challenging attack scenarios. This view fits in with CAF-Net's goal to deliver precise, scalable and context-aware intrusion detection for today's network security applications.

Suthaharan discusses the difficulties in the classification of intrusion network traffic in the Big Data era in his paper [40] where volume, variety and velocity are factors that add to the difficulties. It detests the typical methods implemented in machine learning, which have been used and are not very efficient in handling the dynamic and hyper volume of Big Data. Therefore, it has formulated a new model for the definition of big data based on the concepts of continuum, cardinality and complexity. New and advanced tools, such as representation learning and lifelong learning in predictive network intrusion could be brought to the state-of-the-art tools in the area of Hadoop, Hive, and cloud computing. It also suggests the requirement for the promotion of analytic methods/technologies that can deal with network security efficiently in the Big Data environment.

As shown in Table 1, CAF-Net is distinguished from representative ensemble-based intrusion detection approaches through the simultaneous integration of XGBoost-based bagging, heterogeneous stacking, attention-driven dynamic weighting, and confidence-aware soft voting. Unlike conventional methods that employ isolated ensemble strategies, the proposed framework combines these complementary mechanisms within a unified adaptive architecture, enabling context-sensitive and interpretable intrusion detection.

Table 1: Comparison of CAF-Net with existing ensemble learning approaches.

Method	Bagging	Stacking	Attention	Confidence-Aware Voting	Dynamic Instance Weighting
Random Forest IDS	✓	×	×	×	×
XGBoost IDS	✓	×	×	×	×
Conventional Stacking IDS	×	✓	×	×	×
Attention-Based IDS	×	×	✓	×	✓
Hybrid Ensemble IDS	✓	✓	×	×	×
CAF-Net (Proposed)	✓	✓	✓	✓	✓

3 Proposed Methodology

The flow of the proposed Context-Aware Fusion Network (CAF-Net) with a photograph is shown in Fig. 1 which is a newly designed approach capable of providing an effective and accurate intrusion detection system with NSL-KDD dataset. The proposed framework consists of combining many advanced learning methods, which include bagging with XGBoost, stacking with K Nearest Neighbors (KNN) and Logistic Regression, and an attention-enhanced hybrid meta-learner. In addition to the graphical representation, Algorithm 1 would sequentially narrate the process of CAF-Net, explaining the implementation step-by-step along with the overall workflow. Also, Table 2 would summarize the mathematical formulations that correspond to each phase of the ensemble framework, to allow for a broader understanding of the computations that are involved. Collectively, all these components would provide a well-organized overall detailed description of how the proposed framework is structured and operates.

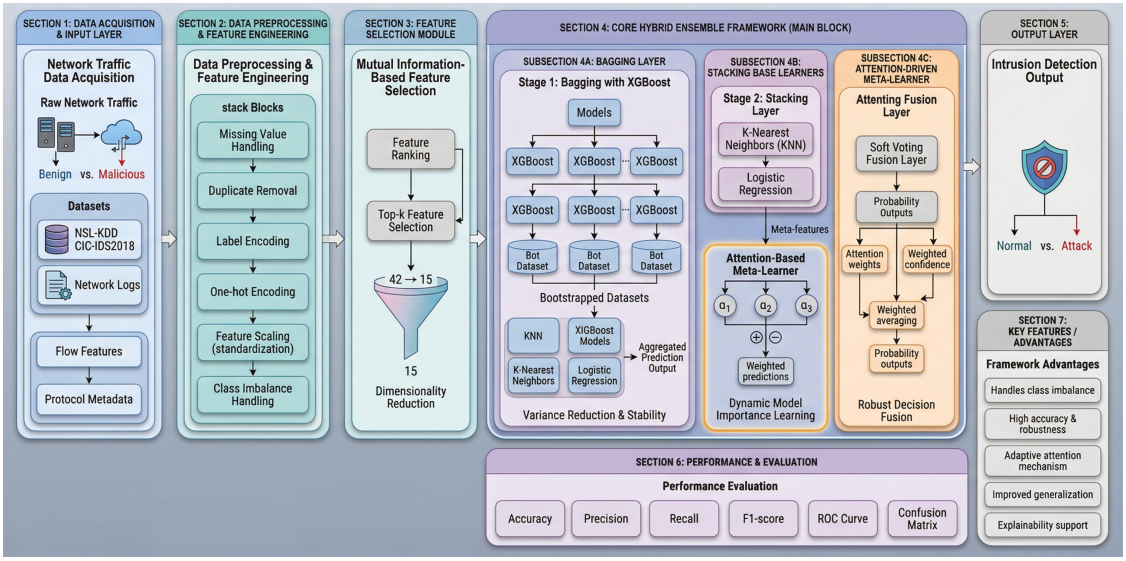


Figure 1: Workflow of proposed methodology.

Table 2: Mathematical representations of proposed context-aware fusion network (CAF-Net).

Component	Description	Mathematical Representation
Stage 1: Bagging	**Technique:** Bagging with XGBoost **Process:** Train multiple XGBoost models on bootstrapped samples. **Objective:** Reduce variance and enhance stability. **Hyperparameters:** $M = 10$, `learning_rate` = 0.1, `max_depth` = 6, `n_estimators` = 100. **Weighted loss function:** $L_{\text{XGB}} = \sum_i \text{weight}_i \cdot \ell(y_i, \hat{y}_i)$, where $\text{weight}_i = 1$ for the majority class and `scale_pos_weight` for the minority class.	Let D be the original dataset. Generate bootstrapped samples: $D_i \sim Bootstrap(D)$ for $i = 1, 2, \ldots, N$ Final prediction from bagging: $\hat{y}_{bagging} = \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i$ where $\hat{y}_i = f_i(D_i)$ (XGBoost predictions)
Stage 2: Heterogeneous Stacking	**Technique:** Stacking with KNN and Logistic Regression. **Process:** Train KNN and Logistic Regression using the predictions produced by Stage 1 bagging as inputs (meta-features). **Hyperparameters:** KNN: $k = 5$, Euclidean, distance-weighted; LR: $C = 1.0$, `class_weight` = `balanced`.	Let $P_{base} = \{\hat{y}_{Bagging}, \hat{y}_{LR}, \hat{y}_{KNN}\}$ be predictions from Bagging, Logistic Regression, and KNN. Intermediate predictions: $\hat{y}_{LR} = f_{LR}(x_i), \quad \hat{y}_{KNN} = f_{KNN}(x_i)$ Meta-feature vector: $Z_i = [\hat{y}_{Bagging}, \hat{y}_{LR}, \hat{y}_{KNN}]^T$
Stage 3: Attention-Based Meta-Learning	**Technique:** Attention network for instance-specific weighting of base-learner predictions. **Process:** Compute attention weights for each base learner and construct attention-weighted meta-features for the meta-learner.	Attention scores computed via feedforward network: **Input:** $Z_i \in \mathbb{R}^{3\times 2}$ (concatenated to $\mathbf{z}_i \in \mathbb{R}^6$). **Hidden layers:** 32 neurons (ReLU, dropout 0.2), 16 neurons (ReLU, dropout 0.2). **Output:** 3 neurons (softmax). $e_j = \left(\mathbf{W}_3\,\text{ReLU}(\mathbf{W}_2\,\text{ReLU}(\mathbf{W}_1\mathbf{z}_i + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3\right)_j$ Normalized attention weights: $\alpha_j = \frac{\exp(e_j)}{\sum_{k=1}^{3}\exp(e_k)}$ Weighted meta-features: $y_{meta} = \sum_j \alpha_j \cdot f_{XGB}(Z_i[j])$
Stage 4: Confidence-Aware Soft Voting	**Technique:** Confidence-aware soft voting with a dual-weighting strategy. **Key distinction:** Dual-weighting integrates instance-specific context (attention) with classifier reliability (trustworthiness).	Trustworthiness scores (validation macro F1): $\beta_j = \text{F1}_j^{\text{val}}, \quad j \in \{\text{Bagging, LR, KNN}\}$ Voting weights: $w_j(x_i) = \alpha_j(x_i) \cdot \beta_j$ Probabilistic aggregation (for $c \in \{0, 1\}$): $P_{\text{final}}(y = c \mid x_i) = \frac{\sum_j w_j(x_i)\cdot P_j(y=c\mid x_i)}{\sum_j w_j(x_i)}$ Final decision: $\hat{y}_i = \arg\max_{c\in\{0,1\}} P_{\text{final}}(y = c \mid x_i)$

Algorithm 1: Proposed context-aware fusion network (CAF-Net)

Input:

$D = \{(x_i, y_i)\}_{i=1}^{N}$: Training dataset, where $x_i \in \mathbb{R}^d$ is the feature vector, $y_i \in \{0, 1\}$ is the class label, and N is the total number of instances.

M: Number of base models for Bagging.

$B = \{f_1^{XGB}, f_2^{XGB}, ..., f_M^{XGB}\}$: Bagging base models (XGBoost models).

$S = \{f^{LR}, f^{KNN}\}$: Stacking base learners, where f^{LR} is Logistic Regression and f^{KNN} is K-Nearest Neighbors.

f_{meta}^{XGB}: XGBoost meta-learner with attention mechanism.

Output: Prediction $\hat{y}_i$ for each instance x_i.

1. Data Preprocessing:

Label Encoding: Convert categorical features $x_{i,j}^{cat}$ into numerical values:

$x_{i,j}^{enc} = \mathcal{L}(x_{i,j}^{cat}), \forall i \in \{1, 2, ..., N\}$.

One-Hot Encoding: Transform categorical feature j into a vector:

$x_{i,j}^{\text{one-hot}} = \mathcal{H}(x_{i,j}^{cat}), j \in \mathcal{C}$.

2. Feature Selection:

Compute mutual information $I(X_j; Y)$ for each feature $j = 1, \ldots, 42$ using:

$I(X_j; Y) = \sum_{x_j} \sum_{y} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)\, p(y)}$.

Rank features by descending $I(X_j; Y)$ and select top $k = 15$ features.

$\mathcal{F} = \{X^{(1)}, X^{(2)}, \ldots, X^{(15)}\}$, where $X^{(1)}$ has the highest MI score.

3. Bagging Phase:

Compute class weight ratio: `scale_pos_weight` $= \frac{\text{count}(y=0)}{\text{count}(y=1)}$.

Generate $M = 10$ bootstrapped datasets $D_1, \ldots, D_M$ from D.

Train an XGBoost model on each D_m with: `learning_rate` $= 0.1$, `max_depth` $= 6$, `n_estimators` $= 100$, `subsample` $= 0.8$, `colsample_bytree` $= 0.8$, and `scale_pos_weight`.

$\hat{y}_{Bagging_i} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_{m,i}^{XGB}, \forall i$.

4. Stacking Phase:

Train Logistic Regression (`C` $= 1.0$, `class_weight` $=$ `balanced`, `solver` $=$ `lbfgs`, `max_iter` $= 1000$) on bagging predictions.

Train KNN ($k = 5$, Euclidean, `weights` $=$ `distance`) on bagging predictions.

Generate meta-feature vector:

$Z_i = \left[P_{Bagging}(y = 1 \mid x_i),\ P_{LR}(y = 1 \mid x_i),\ P_{KNN}(y = 1 \mid x_i)\right]^T \in \mathbb{R}^3$.

5. Attention Network and XGBoost Meta-Learner:

Feed Z_i together with complementary probabilities (concatenated to $\mathbf{z}_i \in \mathbb{R}^6$) into a 3-layer network (32→16→3 neurons, ReLU/softmax).

$e_j = \left(\mathbf{W}_3\, \text{ReLU}(\mathbf{W}_2\, \text{ReLU}(\mathbf{W}_1 \mathbf{z}_i + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3\right)_j,\ j \in \{\text{Bagging, LR, KNN}\}$.

Normalize using softmax:

$\alpha_j = \frac{\exp(e_j)}{\sum_k \exp(e_k)}$.

Compute weighted prediction:

$y_{meta} = \sum_j \alpha_j \cdot f_{XGB}(Z_i[j])$.

(Continued)

Algorithm 1 (continued)

35: **6. Soft Voting Mechanism:**
36: Compute trustworthiness scores from validation set:
37: $\beta_j = \mathrm{F1}_j^{\mathrm{val}},\ j \in \{\text{Bagging, LR, KNN}\}$.
38: Compute combined voting weights:
39: $w_j(x_i) = \alpha_j(x_i) \cdot \beta_j$ for each base learner j.
40: Compute final probabilistic outputs:
41: $P_{final}(y \mid x_i) = \frac{\sum_j w_j(x_i) \cdot P_j(y|x_i)}{\sum_j w_j(x_i)}$.
42: Compute final decision:
43: $\hat{y}_i = \arg\max_y P_{final}(y \mid x_i)$.
44: **7. Return** $\hat{y}_i$

3.1 Dataset

A dataset called NSL-KDD was created to address a number of problems inherent in the original KDD Cup99 dataset by filtering out redundant records and reducing class imbalances [41]. While the NSL-KDD dataset was introduced in 2009 and is now considered a legacy benchmark in the intrusion detection community, it remains widely used as a standardized comparison tool for various intrusion detection techniques due to the scarcity of publicly available labeled network intrusion datasets and the extensive body of prior work that enables consistent comparative evaluation. The dataset may not be an exact replica of modern real-world networks, but its continued use is justified by the need for reproducible benchmarking against established methods in the literature. For CAF-Net training and evaluation, subsets of the NSL-KDD dataset are taken from Kaggle [42], namely the KDDTrain+ and KDDTest+. We classified the labels into two categories—normal and abnormal despite the fact that the dataset originally contains several attack class names, allowing the study to focus on the primary performance measures of binary intrusion detection. Table 3 shows that there are 133,665 total cases in the KDDTrain+ subset, of which 69,908 are classified as normal and 63,757 as abnormal. There are 14,852 total occurrences in the KDDTest+ subset, of which 7146 are normal and 7706 are abnormal [43].

Table 3: Records of subsets for NSL-KDD dataset.

	Total	Normal	Abnormal
KDDTrain$^+$	133,665	69,908	63,757
KDDTest$^+$	14,852	7146	7706

There are 42 features in the dataset, including 41 traffic input features and the class label that indicates whether the traffic record is an attack or a regular packet. The NSL-KDD dataset includes main attacks of four types: DoS, U2R, Probe, and R2L [44]. The 41 traffic features comprise 3 categorical features (`protocol_type`, `service`, `flag`), 6 binary features (`land`, `logged_in`, `root_shell`, `su_attempted`, `is_host_login`, `is_guest_login`), and 32 continuous features (including durations, byte counts, error rates, and various count-based metrics). Several metrics of continued features to capture the important dimensions of network activity are included. These numerical properties are represented by integers and floating-point numbers: connection durations, bytes sent and received, error rates, counts of specific different network activities, service usage rates, server response rates indicating efficiency and timing of the server interactions during the connection. These quantitative statistics should be examined to learn more about the overall flow of traffic from the network, and to identify the possible

presence of any anomalies or suspicious patterns that could signal an intrusion. The categorical features (protocol type, service, and flag) are attributes that have qualitative descriptions to network activities and are very important for capturing accurate states or conditions of network connections. For example, the feature protocol_type tells whether the protocol is TCP, UDP or ICMP. The `service` feature is the network service that is accessed, for example, HTTP or FTP services. The `flag` feature is used to record the status of a connection, including whether the connection was established, in progress or failed, and to record the state of communication at different stages of a network session. When using binary variables, set to `logged_in` means that the login was successful, otherwise the value is `false`. This blend of categorical, binary and continuous data attributes allows for a full and effective analysis of network intrusion detection. Continuous features are used to compute statistical and temporal patterns commonly found in normal and malicious traffic, while categorical and binary features provide context and state-specific information to help build an understanding of the nature of network connections. This multi-type dataset can be used by machine learning models to classify and detect different types of network intrusions efficiently and result in enhanced security breach prevention and mitigation.

3.2 Data Visualization

We included the labels in two distinct categories in our analysis namely, normal and attack as shown in Fig. 2. This lets us determine readily the number of instances of both kinds of network activity. This visualization gives a user an immediate understanding of the traffic distribution, which includes what is considered a normal distribution and threats/attacks. Each category is compared in size and this helps to identify if there may be class imbalance in the data set, for example one type of activity may be much more common than the other. These imbalances can provide important trends that are essential for improving the functioning of IDS systems. For instance, if the traffic shows a high number of normal traffic instances and a low number of attack instances, the models can become biased towards the normal traffic, whereas the opposite should be true, and there should be a reasonably equal number of each type. These visualizations can help network administrators and analysts better understand network traffic patterns and they can help evaluate the effectiveness of the IDSs by counting the number of protocols contained in the dataset [45]. The TCP, UDP, and ICMP protocol types are analyzed and employed for determining the network communication structure based on the characteristics of network traffic. This count plot will give you the information regarding the presence of these protocol types in your data, relative to other normal and attack traffic. Enquiry on the protocol distributions in conjunction with attack categories can offer valuable insights in the relationship between protocol and network threats. For example, if a certain protocol (like UDP) is more likely to be used in malicious or suspicious activities, it may be a weak point to exploit. However, if one had a good understanding of these relationships, targeted network security measures could be designed and in turn, enhance the effectiveness of an IDS.

As we can see in Fig. 3 the data is divided into attacks and no attacks. Visualisation like this can be useful to understand which protocol types are likely to be more or less associated with network threats. The figure, for example, implies that it can be seen trends in the datasets, such as a greater rate of TCP being associated with normal activity, and UDP seemingly more with attack, though these trends need further quantitative validation using statistical analysis or additional experiments. These initial findings could help to inform specific security actions and focus more specific work on strengthening network security where it could be more susceptible to exploitation. We note however that the final conclusions on protocol level attack associations come after careful statistical testing, which are shown as our experimental results, in Section 4.

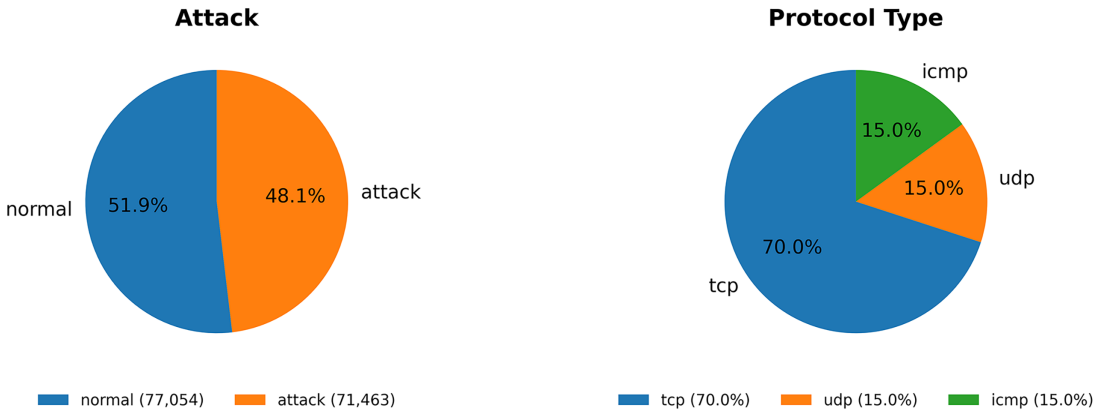


Figure 2: Distribution of network traffic types.

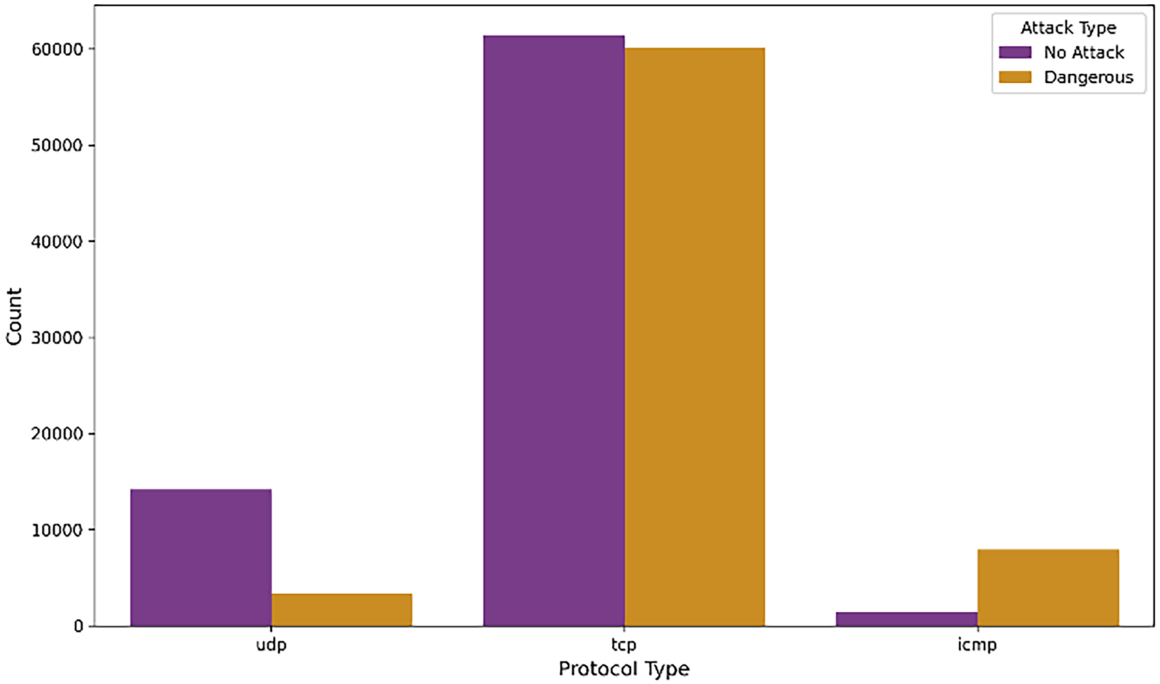


Figure 3: Correlation between protocol types and attack types.

In Fig. 4 we plot the distribution of the normal and abnormal (attack) classes. This plot can be used to examine the comparison of attack instances with normal network traffic in terms of frequency, which provides a more accurate understanding of the composition of this dataset. Visual depiction of this distribution might also assist in understanding whether the distribution of the dataset is skewed, which is a common problem in network intrusion detection datasets. Imbalances must be mitigated by resampling or some algorithmic manipulation of the classifier such that the IDS will not be too biased towards the majority class—which in this case is normal traffic. Proper knowledge of the numbers of normal and abnormal activities will be a good starting point for refining the IDS model, which would improve its ability to detect real-time threats and minimize false positives or false negatives.

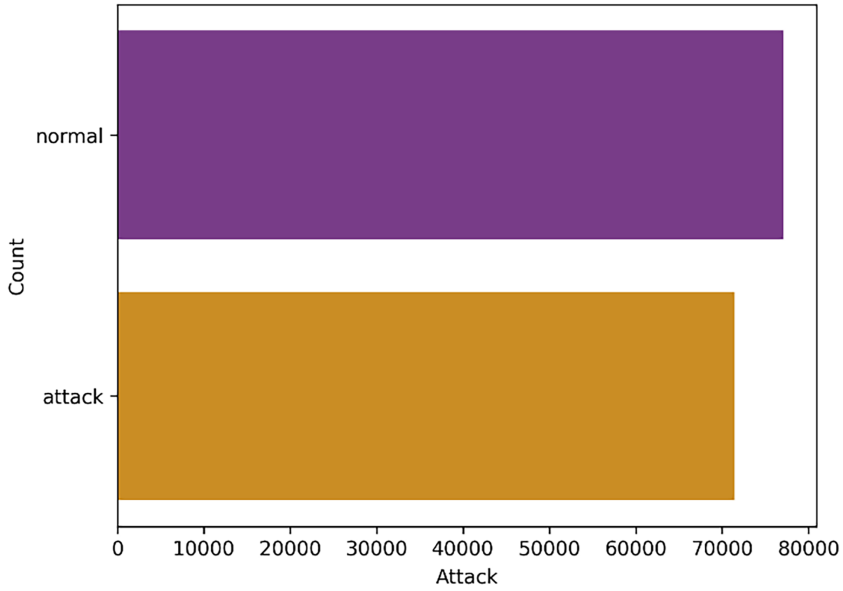


Figure 4: Distribution of count of attacks.

The heat map of the correlation in Fig. 5 was utilized to assess the relations between the attributes chosen in the dataset. The calculation and visualization of the correlation matrix of features demonstrates how features relate to each other; it can be used in feature selection and model design. In our analysis, we plotted the distribution of the duration feature in the various flag categories by using KDE plot. KDE is a continuous approximation of a probability density function.

It takes a step further than histogram and gives a clearer information about the data distribution. The variations in how the distribution of duration varies can be visualized by plotting the KDE curves of each category of flag. This may reveal overlapping distributions or individual peaks, which may not be so clear in discrete histogram forms.

The KDE plot in Fig. 6. give further insight to what is going on underlying the distribution as well as how density differs by category. This is a stacked area density plot that showed the distribution of different network flags over a range of the different connection durations. The x-axis is the "duration" variable, measured in units such as seconds, spreading from 0 to over 50,000, and the y-axis denotes density measured as a number between 0 and 1, showing relative frequency of each flag category at corresponding duration values. Various categories of flags are indicated on the colored regions within the plot, according to the legend on the right side of the page. These categories comprise "SF" (blue), "S0" (light pink), "REJ" (yellow), "RSTR" (green), "SH" (light purple), "RSTO" (grey), "S1" (black), "RSTOS0" (dark purple), "S3" (teal), "S2" (light yellow), and "OTH" (light blue). It can be easily inferred from the plot that the majority flag is "S0" which is colored light pink particularly over the mid to high duration values. Thus, this flag appears very often for longer connection durations. On the other hand, the "SF" flag is colored blue, meaning that it has a higher density in a short duration of connections; but its appearance decreases substantially if the duration becomes larger. Other flags, like "REJ," "RSTR," etc. tend to have lower densities. They spread sparsely over the range of duration and occasionally show peaks where they may be suggesting increased occurrence. This visualization can thus be very fruitful in how different flags behave with respect to the connection duration, thus showing certain patterns, which may be crucial in understanding network behavior or anomaly detection or traffic classification in cyber security terms.

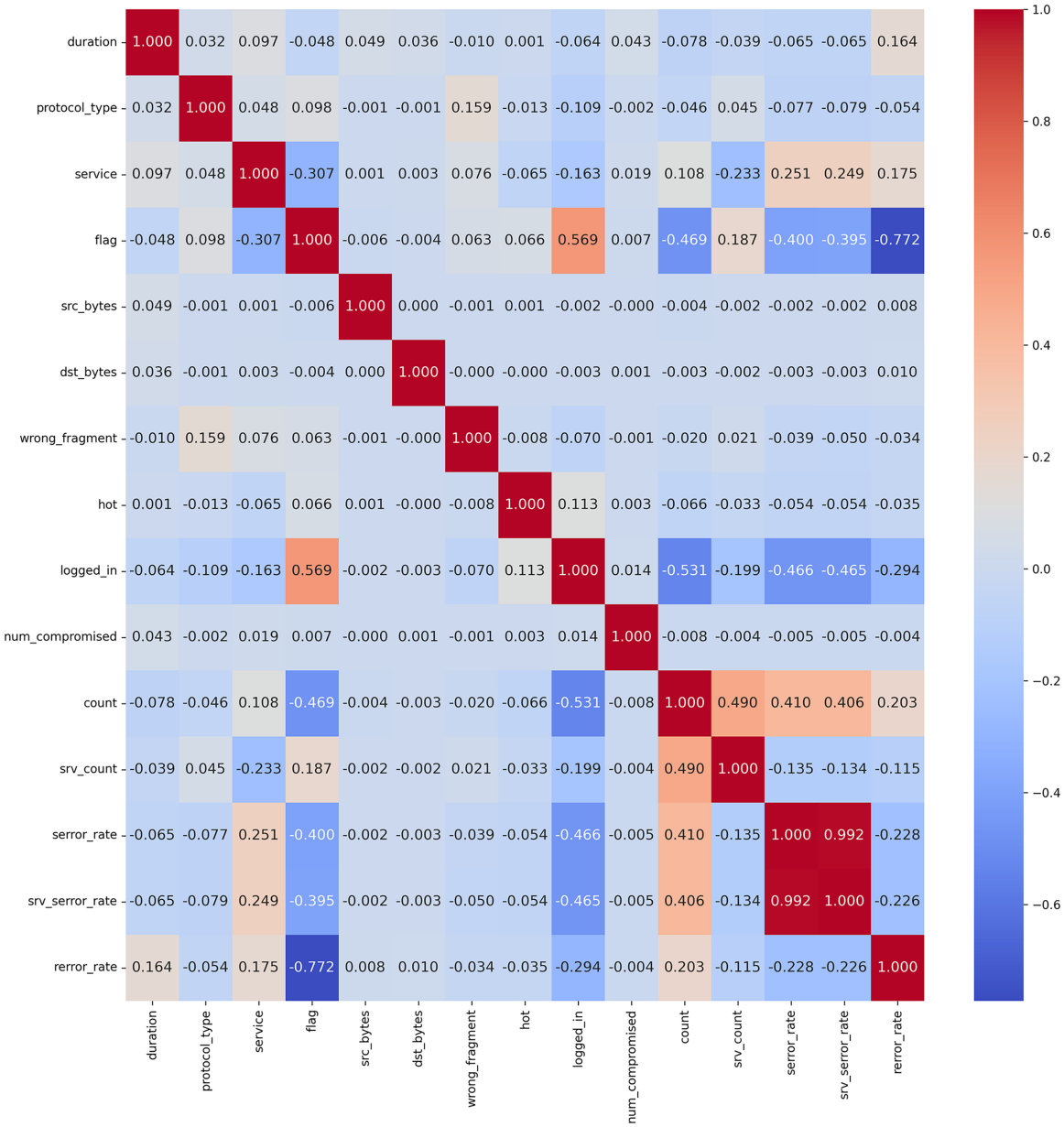


Figure 5: Correlation between selected features.

Boxplots help to show a short summary of the distributions of the features, showing median, quartiles and possible outliers, and also are useful for making some comparisons between groups or subsets of data. They are especially helpful in identifying outliers and determining their possible effect on the data set. Boxplots of these selected features are shown in Fig. 7 and no outliers (defined as data values more than 1.5 times the interquartile range from the quartiles on either side of the median) are observed. The values of the whiskers are inside the range of all the values, so most of the values are inside the interquartile range. Moreover, there are no values outside the whiskers, which indicates that the distributions of features are not highly skewed and there are no outliers that need special consideration. By this analysis, we found that the explicit removal of outliers was not necessary as the deviations are not meaningful in terms of interpretation of results or a meaningful impact on model performance.

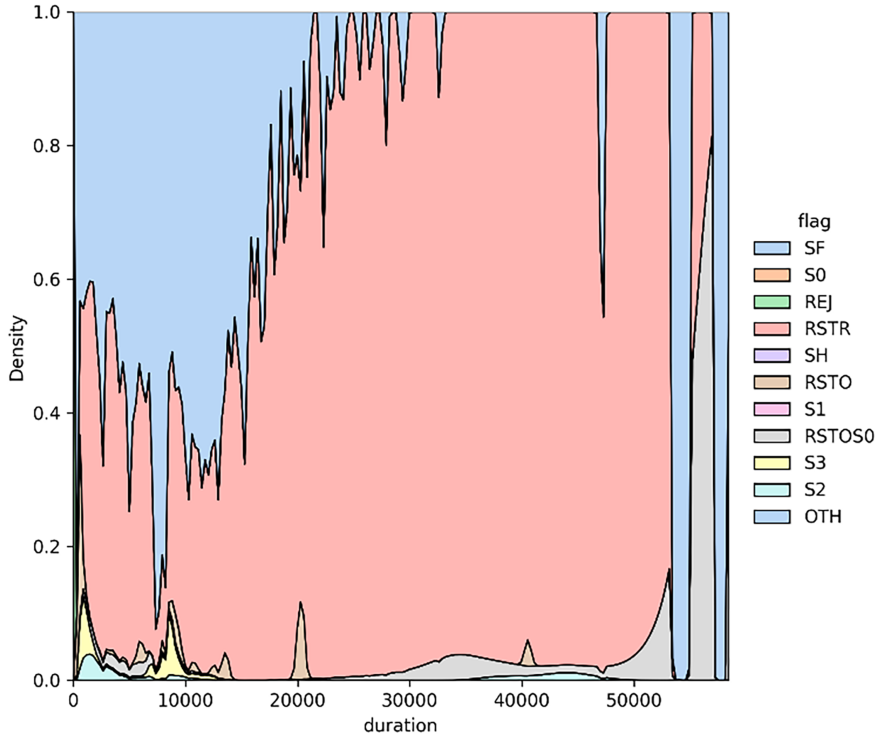


Figure 6: Kernel density estimate plot of duration by flag.

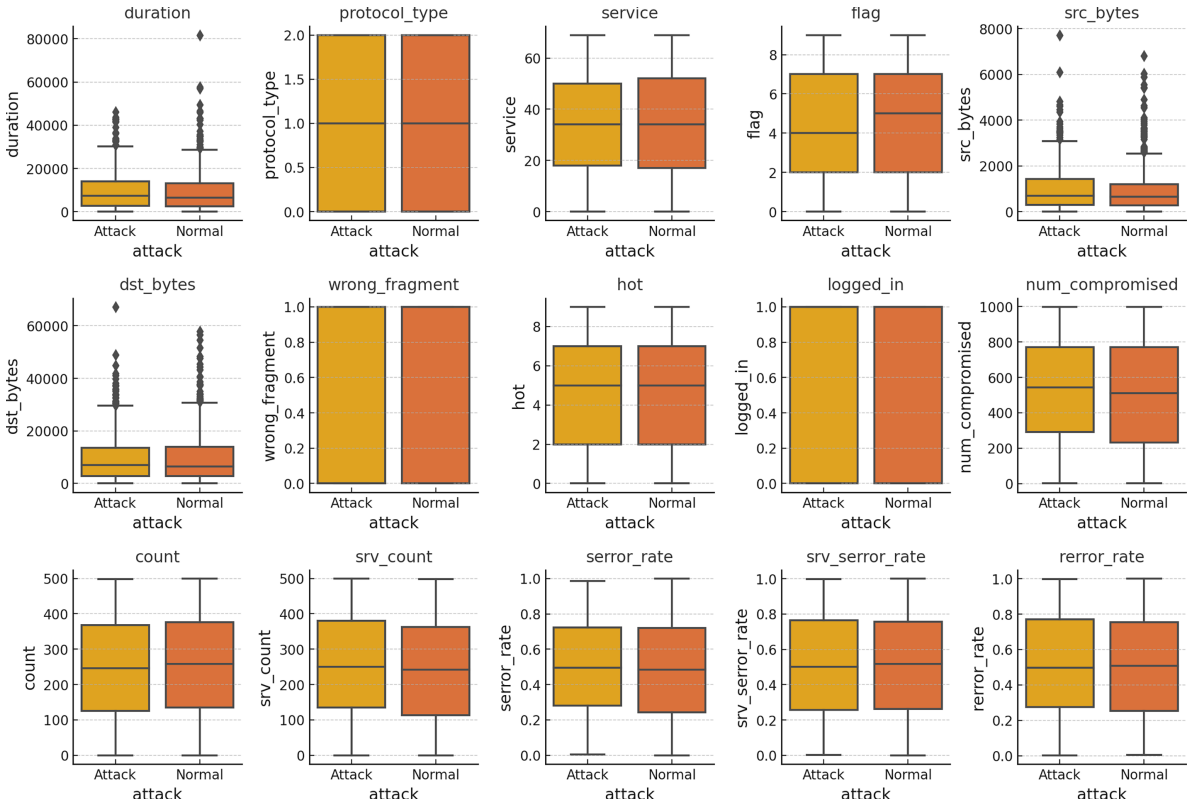


Figure 7: Box plots for selected feature.

For better analysis, also to reduce the size of the set that is still preserving the variations in the set, we used the PCA technique. PCA is a good method to perform the dimensionality reduction of data into a lower number of principal components in directions of maximum data variance. The first two principal components of this dataset explain a large portion of the variation and these turn out to be fundamental in retaining the structural information from the high-dimensional original space.

The scatter plot shows this separation between the classes, which is essential to find the structure and patterns of the relationships between the classes. This visualization shows that there are class differences in the data set, which means that machine learning models may be able to learn discriminative patterns that can be correctly classified. The distinct separation of the "Normal" class and the "Attack" class shows the usefulness of PCA in both dimensionality reduction and visualising the class distributions. The relationships are evident on the 2D plot created from the first two principal components and this allows us to obtain an intuitive understanding of the dataset's structure. Before this PCA-based reduction is used here, it is important to note that this reduction is employed for the purpose of visualising and exploratory analysis only, and is not actually a variable to be used in the classification model. In the actual model pipeline, the top 15 most discriminative features are selected from the original 42 features for the model, by making use of mutual information-based feature selection (Section 3.4).

Fig. 8 shows the graphical interpretation of the NSL-KDD dataset by applying PCA. In the KDDTrain+ dataset, the distribution of normal and abnormal samples is visualized. The quantity of traffic input for both normal and abnormal conditions appears to be relatively balanced in this subset. The normal samples form a distinct cluster, while the abnormal samples exhibit three separate clusters, suggesting that the features of each label class in the traffic input samples are distinguishable. Fig. 9 displays the distribution of normal and abnormal samples in the KDDTest+ dataset. The number of abnormal samples surpasses that of the normal samples in this test subset. The characteristics of the abnormal traffic samples appear to be more heterogeneous, as evidenced by the lack of any clear cluster formation nearby [20].

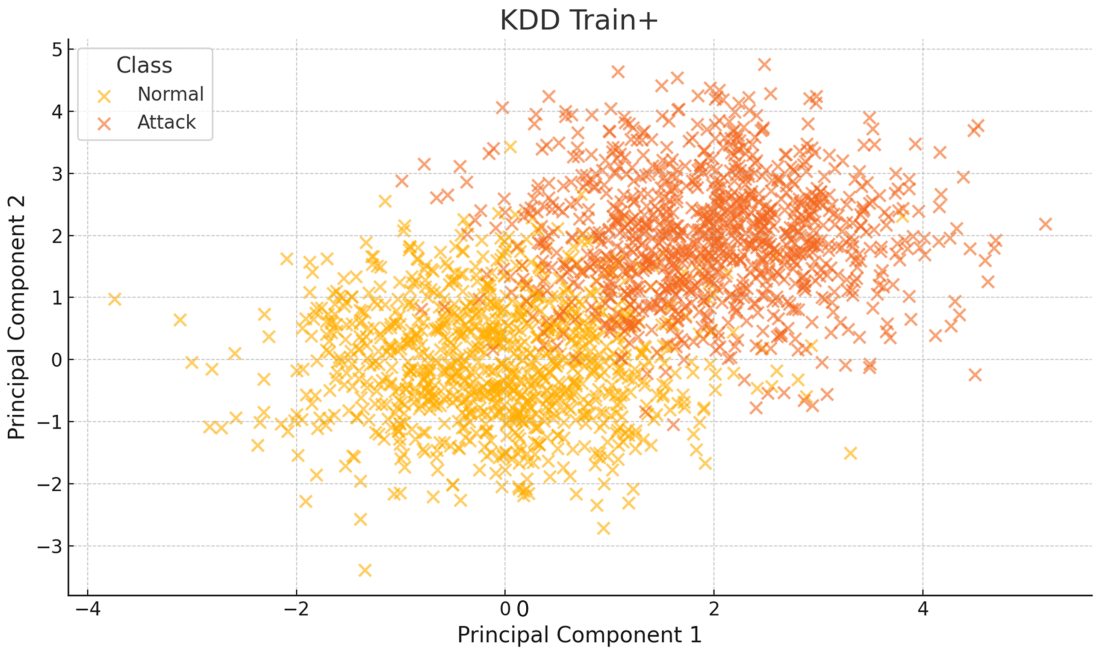


Figure 8: Visualization of PCA for NSL-KDD subset KDDTrain+.

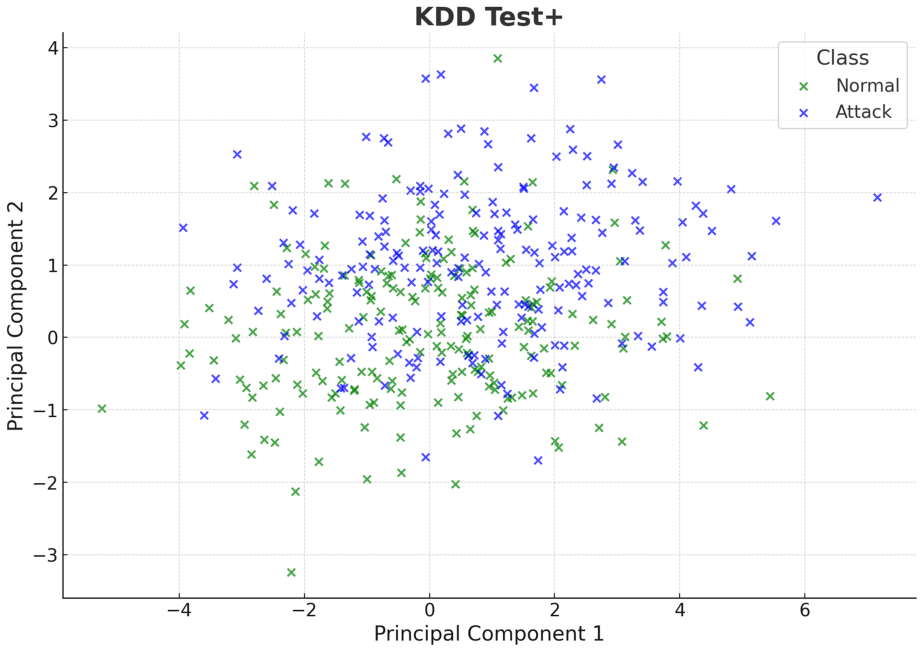


Figure 9: Visualization of PCA for NSL-KDD subset KDDTest+.

3.3 Data Pre-Processing

Subsets like KDDtrain+ and KDDTest+, which is a part of the NSL-KDD dataset, is the primary source used in our research. For training and validation, we employed KDDtrain+ and KDDTest+. NSL-KDD contains an adequate number of records in its train and test sets [42]. Upon loading the Features such as duration, protocol type, application (service), and traffic metrics, among other data, are stored preprocess it meticulously. This includes specifying the type of protocol and process type as categorical variables and specifying the n-aforementioned. They had to be converted into numbers using LabelEncoder for them to be usable in machine learning algorithms [46]. Furthermore, we classify attacks as a binary classification problem, 'normal' and 'attack', to ease the supervised learning problems that are essential for the model training and evaluation in the subsequent step.

- **NSL-KDD:** NSL-KDD: This dataset comes with the training and testing subsets preconfigured: KDDTrain+ (133,665 instances) for training and KDDTest+ (14,852 instances) for testing. The split of this standard was adopted to be comparable with previous studies [20,22,24]. The training-test splitting of the dataset is provided for each dataset and the predefined training-test split (`UNSW-NB15_training-set.csv` and `UNSW-NB15_testing-set.csv`) was used according to standard evaluation procedures in the literature.
- **CIC-IDS2018:** The data set does not have a pre-defined split. Thus, 80/20 split using stratified random sampling (80% training and 20% testing) was performed to maintain the class distribution in both split sets. This split was used for all models.

For all datasets, a stratified sampling approach was used during splitting to maintain the original class distribution in both training and test sets, ensuring representative evaluation of the minority abnormal (attack) class.

3.4 Feature Engineering

Not all features in the NSL-KDD dataset contribute equally to the prediction of network intrusions. Some features may be redundant or exhibit low discriminative power, which can increase dimensionality,

lengthen processing time, and potentially degrade model performance in certain scenarios. Enhancing our models' predictive power therefore requires effective feature engineering techniques. We employ mutual information (MI)-based feature selection to identify, select, and apply the most relevant features from the dataset. MI quantifies the dependence of each feature X_j with the target variable Y using the following computation:

$$I(X_j; Y) = \sum_{x_j \in \mathcal{X}_j} \sum_{y \in \mathcal{Y}} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)\, p(y)},$$

The term $I(X_j; Y)$ quantifies the reduction in uncertainty about the class Y (i.e., the shared information) given knowledge of the feature X_j. The features are sorted according to the MI scores which are displayed in Table 4 and plotted in Fig. 10 from the largest to the smallest.

Table 4: Feature scores of traffic features from the NSL-KDD dataset ranked by mutual information.

Features	Score
src_bytes	0.565883
service	0.469267
dst_bytes	0.440655
flag	0.367972
same_srv_rate	0.366969
diff_srv_rate	0.361037
dst_host_srv_count	0.335818
dst_host_same_srv_rate	0.310648
logged_in	0.292594
dst_host_serror_rate	0.285128
dst_host_diff_srv_rate	0.283794
dst_host_srv_serror_rate	0.281094
serror_rate	0.277144
srv_serror_rate	0.268365
count	0.264521
dst_host_srv_diff_host_rate	0.184886
dst_host_count	0.144783
dst_host_same_src_port_rate	0.132819
srv_diff_host_rate	0.096355
srv_count	0.063776
dst_host_srv_rerror_rate	0.061091
protocol_type	0.055061
rerror_rate	0.039194
dst_host_rerror_rate	0.037665
num_outbound_cmds	0.000000
su_attempted	0.000000
land	0.000000
is_guest_login	0.000000

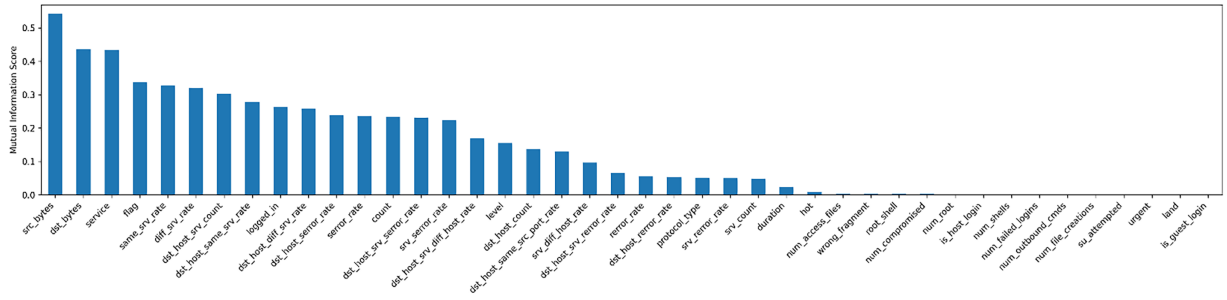


Figure 10: Features sorted based on mutual information score.

It is important to note that the NSL-KDD dataset contains 42 total features (41 traffic features and 1 class label). However, not all features contribute equally to the prediction of network intrusions. To improve model efficiency, reduce overfitting risk, and enhance interpretability, we employ a feature selection strategy. A top-k feature selection strategy based on Mutual Information (MI) was employed, whereby the 15 highest-ranked features (k = 15) were retained for subsequent model training. The value of k = 15 was found using the elbow plot of the MI scores (Fig. 10) where all of the information gain seems to level off a bit, meaning that the additional features will make only slight improvements in the predictive power. This approach provides an optimal trade off between dimension reduction, computational efficiency and information retention. Table 4 shows the top rankings of traffic features on NSL-KDD, and Table 5 shows the top 15 traffic features chosen for implementing the proposed CAF-Net framework.

Table 5: Selected features from the NSL-KDD dataset.

No.	Selected Feature
1	src_bytes
2	service
3	dst_bytes
4	flag
5	same_srv_rate
6	diff_srv_rate
7	dst_host_srv_count
8	dst_host_same_srv_rate
9	logged_in
10	dst_host_serror_rate
11	dst_host_diff_srv_rate
12	dst_host_srv_serror_rate
13	serror_rate
14	srv_serror_rate
15	count

3.5 Feature Scaling

Feature scaling is the process of scaling the data so that each feature has mean 0 and standard deviation 1 [47]. To ensure consistency across the features and boost the reliability of the model, standardization was done using 'StandardScaler'. This transformation centers each feature by subtracting its average from it, and normalizes it, or divides by the standard deviation. This way, differences in feature scales do not

over-impact there learning process. Additionally, by standardizing the data, the proposed model can be fairly and accurately evaluated. First, the scaler was only fit on the training set; then both training and test were transformed with the fitted scaler, as is standard in order to avoid information leakage. This means that information from the test set does not affect the scaling parameters, which in turn does not affect the results of the test and which will be unbiased to evaluate the performance of the model.

$$\mathbf{Z} = \frac{(x - \mu)}{\sigma} \tag{1}$$

In Eq. (1), the variable z is the standardized value, the feature value is x, the mean of the feature is μ and the standard deviation is σ.

3.5.1 Class Imbalance Handling Strategy

NSL-KDD, UNSW-NB15 and CIC-IDS2018 datasets are very imbalanced in some attack categories with significantly higher occurrence of normal traffic instances than attacks. To overcome this challenge, CAF-Net adopts a multi-facet approach combining weighted learning mechanisms at various levels of the ensemble framework without referring to modifying the original data distribution.

1. **Weighted loss function in bagging with XGBoost:** During the bagging phase, each XGBoost base model is configured with the `scale_pos_weight` parameter, defined as

$$\texttt{scale\_pos\_weight} = \frac{\text{count}(y = 0)}{\text{count}(y = 1)}, \tag{2}$$

where $\text{count}(y = 0)$ and $\text{count}(y = 1)$ denote the number of normal and attack instances, respectively, in the training set.

2. **Class-weighted logistic regression:** In the stacking phase, Logistic Regression is trained using `class_weight = balanced`, which assigns weights inversely proportional to class frequencies:

$$\text{weight}_c = \frac{N}{K \cdot \text{count}(y = c)}, \tag{3}$$

where N is the total number of instances, $K = 2$ is the number of classes, and $\text{count}(y = c)$ is the number of samples in class c.

3. **Distance-weighted KNN:** The KNN classifier employs distance-weighted voting,

$$w_k = \frac{1}{d_k + \epsilon}, \tag{4}$$

where d_k is the distance to the k-th nearest neighbor and ϵ is a small constant to avoid division by zero.

4. **Trustworthiness-aware soft voting:** The confidence-aware soft voting mechanism contributes to imbalance handling by using trustworthiness scores β_j based on validation macro F1-score, which is sensitive to minority-class performance. Consequently, base learners that better detect attack instances receive higher voting weights in the final aggregation.

5. **Attention-driven instance weighting:** The attention-based meta-learner dynamically computes instance-specific importance weights for base-learner predictions based on the characteristics of the current input instance. This enables adaptive, context-sensitive decision-making that can benefit instances across both majority and minority classes, without explicitly targeting a particular class. The attention

mechanism operates independently of the class label, as described in Section 3.6, computing weights solely from the probabilistic outputs of the base learners.

Design rationale for avoiding resampling: The design reasons for not explicitly performing resampling (such as SMOTE, random oversampling, or undersampling): We deliberately do not explicitly use resampling methods (SMOTE, random oversampling or undersampling) to retain data integrity, to avoid computational costs and to avoid changing the distribution of the data in the ensemble.

This multi-pronged approach is shown to be effective in the ablation study presented in Section 4.2, where implementation of the complete CAF-Net model outperforms models that lack attention mechanisms and soft voting.

3.6 CAF-Net: Proposed Framework

Traditional ensemble learning techniques for intrusion detection generally adopt a fixed method of assigning weights to the individual classifiers, majority voting, or single bagging/stacking architectures. These techniques are useful in some cases, but they may not always be able to cope with the unpredictable and varied behavior of network traffic. To overcome these drawbacks, proposed Context-Aware Fusion Network (CAF-Net) combines the XGBoost-based bagging, heterogeneous stacking approaches with KNN and Logistic Regression, attention-based meta-learning, and a confidence-aware soft voting mechanism, as shown in Fig. 11. The proposed CAF-Net is a four-stage ensemble learning framework tailored to achieve high accuracy and interpretability: Stage 1 applies bagging with XGBoost to reduce variance and improve stability, Stage 2 performs heterogeneous stacking (KNN + LR) to capture complementary local and global decision patterns, Stage 3 introduces attention-based meta-learning to assign instance-specific importance to base-learner predictions, and Stage 4 performs confidence-aware soft voting to fuse decisions using both attention relevance and model trustworthiness. This context-aware fusion is dynamically adjusted to the content of every traffic instance based on its characteristics for the context-aware IDS.

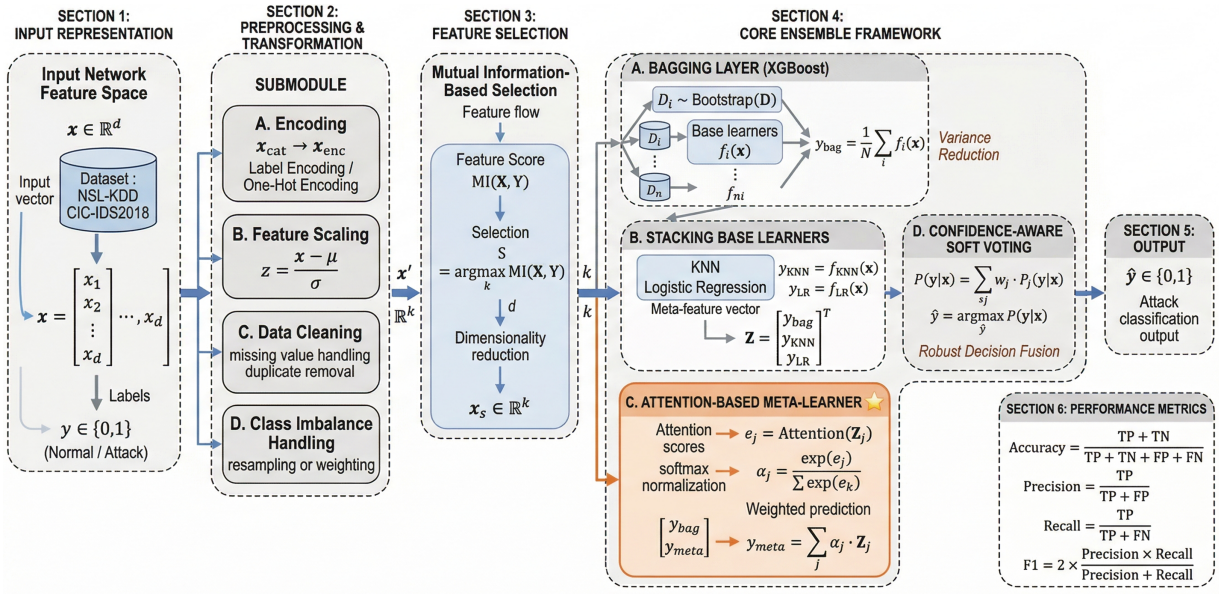


Figure 11: Proposed context-aware fusion network (CAF-Net) approach.

3.6.1 Architecture and Hyperparameter Configuration

The CAF-Net framework follows a systematic pipeline with clearly defined stages and hyperparameters. Below we provide the complete architectural specification to ensure reproducibility.

Pipeline Order:

1. **Data Preprocessing:** Label encoding for categorical features (`protocol_type`, `service`, `flag`), followed by standardization using `StandardScaler` (zero mean, unit variance).
2. **Feature Selection:** Mutual information-based top-15 feature selection (see Section 3.4).
3. **Stage 1—Bagging (XGBoost):** Train $M = 10$ XGBoost models on bootstrapped samples of the training data.
4. **Stage 2—Stacking (KNN + Logistic Regression):** Train KNN and Logistic Regression on the bagging predictions.
5. **Stage 3—Attention-Based Meta-Learning:** Train the attention network jointly with the XGBoost meta-learner.
6. **Stage 4—Confidence-Aware Soft Voting:** Aggregate final predictions using dual-weighting.

Hyperparameter Configuration: The full configuration is summarized in Table 6.

Training Protocol: The training process follows a two-phase protocol to prevent information leakage and ensure generalization:

1. **Phase 1 (Base-learner training and validation split):** The training portion of each dataset is further divided into training (80%) and validation (20%) subsets using stratified sampling to preserve the original class distribution. The training subset is used for learning the parameters of the Stage 1 and Stage 2 learners (bagging with XGBoost, KNN, and Logistic Regression), while the validation subset is used for hyperparameter tuning and early stopping.
2. **Phase 2 (Meta-learning and fusion training):** The Stage 3 attention network and the associated meta-learner are trained using the validation predictions generated by the base learners, and validation performance is also used to compute the trustworthiness scores required by the Stage 4 confidence-aware soft voting fusion. This protocol prevents information leakage and improves the generalization capability of the ensemble framework.

Final Evaluation: The trained framework is evaluated on the test set, which is held out from the entire training process.

This framework has the strength of being dynamic in the sense that it prioritizes the role of individual base learners based on attention without compromising the end prediction that can be explained. Not only does this help counter the limitations of conventional IDS (e.g., low accuracy and manual feature engineering), but it also offers a new frame of reference of functionality and interpretability of intrusion detection.

XGBoost bagging is used in Stage 1 to ensure that predictions are more stable with lower variance. In this stage, multiple XGBoost models are trained on bootstrapped samples of the training data. The models produce diverse predictions that are incorporated as inputs in the subsequent stages. This approach enables the model to capture a wide range of patterns in the data, prevents overfitting, and improves generalization. The bagging predictions are then passed to Stage 2, where KNN and Logistic Regression serve as base learners. These learners generate intermediate predictions, which are concatenated with the bagging predictions to form a 3-dimensional meta-feature vector $Z_i = [P_{Bagging}(y = 1|x_i), P_{LR}(y = 1|x_i), P_{KNN}(y = 1|x_i)]^T$. It is expected that stacking heterogeneous models will reflect both local (KNN) and global (Logistic Regression) decision patterns in the data. Notably, no fusion into a single representation occurs at Stage

2; rather, the meta-feature vector is passed to Stage 3, where the attention-based meta-learner performs the actual adaptive weighting and fusion. Stage 3 involves, as one of the main novelties of our model, a hybrid meta-learner with an attention mechanism that dynamically computes instance-specific importance weights for the base-learner predictions. The final decision fusion is then performed in Stage 4 through the confidence-aware soft voting mechanism.

Table 6: CAF-Net architecture and hyperparameter configuration.

Component	Parameter	Value
XGBoost Bagging	Number of base models (M)	10
XGBoost Bagging	Learning rate	0.1
XGBoost Bagging	Maximum depth	6
XGBoost Bagging	Number of boosting rounds	100
XGBoost Bagging	Subsample ratio	0.8
XGBoost Bagging	Column subsample ratio	0.8
XGBoost Bagging	`scale_pos_weight`	$\frac{\text{count}(y=0)}{\text{count}(y=1)}$
XGBoost Bagging	Early stopping patience	10
KNN	Number of neighbors (k)	5
KNN	Distance metric	Euclidean
KNN	Weighting	Distance-weighted
Logistic Regression	Regularization (C)	1.0
Logistic Regression	Class weight	`Balanced`
Logistic Regression	Solver	`lbfgs`
Logistic Regression	Max iterations	1000
Attention Network	Input dimension	6 (3×2 probabilities)
Attention Network	Hidden layer 1	32 neurons, ReLU, dropout 0.2
Attention Network	Hidden layer 2	16 neurons, ReLU, dropout 0.2
Attention Network	Output layer	3 neurons, softmax
Attention Network	Optimizer	Adam (lr = 0.001)
Attention Network	Batch size	128
Attention Network	Epochs	100 (early stopping patience 10)
Attention Network	Weight initialization	Xavier uniform
Meta-Learner (XGBoost)	Learning rate	0.05
Meta-Learner (XGBoost)	Maximum depth	4
Meta-Learner (XGBoost)	Boosting rounds	50
Meta-Learner (XGBoost)	`scale_pos_weight`	$\frac{\text{count}(y=0)}{\text{count}(y=1)}$
Soft Voting	Trustworthiness weighting	$\beta_j = \text{F1}_j^{\text{val}}$
Soft Voting	Combined weight	$w_j(x_i) = \alpha_j(x_i) \cdot \beta_j$

Input representation: For each instance x_i, the attention network receives a 6-dimensional input vector $\mathbf{z}_i \in \mathbb{R}^6$, formed by concatenating the probabilistic predictions $[P_j(y=0),\ P_j(y=1)]$ from each of the three base learners.

Network architecture: The attention network consists of three fully connected layers:

- **Hidden layer 1:** 32 neurons with ReLU activation and dropout rate $p = 0.2$.
- **Hidden layer 2:** 16 neurons with ReLU activation and dropout rate $p = 0.2$.
- **Output layer:** 3 neurons with softmax activation, producing normalized attention weights α_j that sum to 1 across the three base learners.

Mathematical formulation: The attention weights are computed as

$$\mathbf{e} = \mathbf{W}_3\,\mathrm{ReLU}\big(\mathbf{W}_2\,\mathrm{ReLU}(\mathbf{W}_1\mathbf{z}_i + \mathbf{b}_1) + \mathbf{b}_2\big) + \mathbf{b}_3, \tag{5}$$

$$\alpha_j = \frac{\exp(e_j)}{\sum_{k=1}^{3}\exp(e_k)}, \quad j \in \{\text{Bagging, LR, KNN}\}, \tag{6}$$

where $\mathbf{W}_1 \in \mathbb{R}^{32\times 6}$, $\mathbf{W}_2 \in \mathbb{R}^{16\times 32}$, and $\mathbf{W}_3 \in \mathbb{R}^{3\times 16}$ are learnable weight matrices, and $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ are the corresponding bias vectors.

Training objective: The attention network is trained jointly with the XGBoost meta-learner using binary cross-entropy loss

$$\mathcal{L}_{\text{meta}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=0}^{1} y_{i,c}\log\left(\hat{y}_{i,c}^{\text{meta}}\right), \tag{7}$$

where $\hat{y}_{i,c}^{\text{meta}}$ denotes the meta-learner's predicted probability for class c after applying attention-weighted aggregation of base-learner predictions.

Training strategy: The Adam optimizer is used with a learning rate of 0.001 and batch size of 128, and a maximum of 100 epochs as training time, and an early stopping parameter of patience 10, which is used to prevent overfitting. Network parameters are initialized with Xavier uniform initialization, and training is done on a subset of data which is designated as validation data to make sure the model is generalized.

This attention-based weighting makes the meta-learner focus on the most informative predictions for each instance and achieve more adaptive and robust decision making. For instance, for certain attack types, KNN might make more accurate predictions, and the attention mechanism would give more weight to the corresponding prediction, thus helping the meta-learner focus more on this information in the final prediction. In the aggregation phase, a soft voting scheme with a confidence level is applied to make the robustness and interpretability of the framework more meaningful. Instead of using a single weighting dimension or simply applying uniform weights to all base learner predictions, our idea is based on a dual-weighting strategy which combines two complementary sources of information: (i) attention-derived base relevance scores $\alpha_j(x_i)$, indicating the relevance of the prediction for the specific instance x_i; and (ii) scores of the trustworthiness of the model β_j based on the validation set accuracy of a base learner j.

Trustworthiness score calculation: For each base learner $j \in \{\text{Bagging, LR, KNN}\}$, the trustworthiness score β_j is computed as its macro F1-score on a held-out validation set:

$$\beta_j = \mathrm{F1}_j^{\text{val}} = 2 \times \frac{\text{Precision}_j^{\text{val}} \times \text{Recall}_j^{\text{val}}}{\text{Precision}_j^{\text{val}} + \text{Recall}_j^{\text{val}}}. \tag{8}$$

These scores are computed once during training and remain fixed during inference, providing a measure of each classifier's overall capability.

Final voting weight calculation: The combined voting weight $w_j(x_i)$ for base learner j on instance x_i is computed as

$$w_j(x_i) = \alpha_j(x_i)\cdot\beta_j, \quad j \in \{\text{Bagging, LR, KNN}\}. \tag{9}$$

Probabilistic aggregation: The final predicted probability for class $c \in \{0,1\}$ is obtained through weighted averaging:

$$P_{\text{final}}(y = c \mid x_i) = \frac{\sum_{j \in \mathcal{B}} w_j(x_i) \cdot P_j(y = c \mid x_i)}{\sum_{j \in \mathcal{B}} w_j(x_i)}, \tag{10}$$

where $\mathcal{B} = \{\text{Bagging, LR, KNN}\}$ is the set of base learners, and $P_j(y = c \mid x_i)$ denotes the predicted probability from base learner j for class c.

Final decision: The final class prediction is determined by

$$\hat{y}_i = \arg\max_{c \in \{0,1\}} P_{\text{final}}(y = c \mid x_i). \tag{11}$$

Algorithmic summary: The soft voting process can be summarized as follows in Algorithm 2.

Algorithm 2: Confidence-aware soft voting

Input: Base learner predictions $P_j(y \mid x_i)$ for $j \in \{\text{Bagging, LR, KNN}\}$; attention scores $\alpha_j(x_i)$; trustworthiness scores β_j.
Output: Final prediction $\hat{y}_i$.
For each test instance x_i:
(1) Compute voting weights: $w_j = \alpha_j(x_i) \cdot \beta_j$ for $j \in \{\text{Bagging, LR, KNN}\}$.
(2) Compute weighted probabilities: $P_{\text{final}}(y = 1 \mid x_i) = \frac{\sum_j w_j P_j(y=1|x_i)}{\sum_j w_j}$ and $P_{\text{final}}(y = 0 \mid x_i) = 1 - P_{\text{final}}(y = 1 \mid x_i)$.
(3) Return $\hat{y}_i = \arg\max_{c \in \{0,1\}} P_{\text{final}}(y = c \mid x_i)$.

The goal of such a dual-weighting scheme is to have the final decision based on the relevance of the instance (given by the attention score) and on the general trustworthiness of the base learner (provided by the trustworthiness score). This design is especially useful in intrusion detection systems that have various classifiers that detect different types of attacks, and in which a different weight is optimal for different samples of network traffic.

Our model is new, because of the use of more advanced techniques (such as attention based weighting techniques, soft voting with confidence scores, heterogeneous ensemble learning). These innovations were added to make a robust highly comprehensible and reliable IDS, and were quite a milestone in the intrusion detection field. In addition to enhancing the state of the art in machine learning-based IDS, the work also offers a platform, based on which future studies on explainable AI and adaptive cybersecurity systems could be conducted.

To conclude, the main novelty of CAF-Net is that it is a single framework integrating dynamic attention based on weighting, confidence-aware soft voting and a balanced bagging-stacking ensemble learning approach in one. CAF-Net not only takes into account the reliability of classifiers but also dynamically adjusts the contribution of individual classifiers based on the properties of each network traffic instance as opposed to the traditional ensemble method which uses a static aggregation scheme. This integration allows for more flexible, understandable and powerful intrusion detection of various attack scenarios.

3.7 Evaluation Metrics

The effectiveness of the proposed approach is assessed using several performance metrics derived from the outcomes of the training and testing phases of the dataset. These metrics are computed based on the classification results and are defined as follows:

1. True Positive (TP): Correct identification of an unauthorized or intrusive user.
2. False Positive (FP): A legitimate user incorrectly classified as an intruder.
3. True Negative (TN): Correct classification of a legitimate user.
4. False Negative (FN): An intrusive user incorrectly labeled as a legitimate one.

To evaluate the performance of the proposed CAF-Net framework, this study considers accuracy along with precision, recall, and the F1 score. These metrics are widely used in network traffic anomaly detection to provide a comprehensive evaluation of model performance.

Accuracy—Accuracy measures the ability of a classifier to correctly identify both normal and intrusive instances in the dataset. It is defined as the ratio of correctly classified samples to the total number of samples [48]. The accuracy calculation is expressed in Eq. (12).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{12}$$

Precision—Precision represents the proportion of correctly predicted intrusive instances among all instances classified as intrusive [48]. The mathematical expression for precision is shown in Eq. (13).

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

Recall—Recall refers to the ability of the model to correctly identify all actual intrusion instances within the dataset [4]. It is computed using Eq. (14).

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

F1 Score—The F1 score is a composite metric that balances precision and recall, providing a unified measure of a model's detection capability [49]. It is particularly significant in intrusion detection systems and is calculated as shown in Eq. (15).

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{15}$$

4 Results and Discussions

In order to see how well these machine learning algorithms perform the NSL-KDD dataset has been used to perform a strict cross validation. To acquire a wider perspective of the performance of the proposed CAF-Net framework, ROC analysis and confusion matrix were used. The ROC curve shows the effect of different decision thresholds on the True Positive Rate (TPR) and False Positive Rate (FPR); it represents the overall power and stability of the trained model. The confusion matrix, on the other hand, provides a succinct and unambiguous overview of the classification performance, showing the numbers of true successes, true failures, false successes and false failures, and the model's ability to distinguish between the classes. This analysis makes results easier to understand, allowing performance to be assessed, and areas of improvement are easy to spot [50].

The proposed CAF-Net framework is shown in a confusion matrix as shown in Fig. 12. It shows how well the classification is working in terms of true positive (TP), true negative (TN), false positive (FP) and false negative (FN). The model was able to correctly classify 7110 normal instances and 7667 attack instances, with only 39 samples of an attack class being misclassified as normal, and 36 samples of a normal class being misclassified as attack samples for a total of 14,852 evaluated test instances (7146 of which were normal and 7706 of which were attack instances). Overall, the proposed framework achieved an accuracy of 99.47%, precision of 99.47%, recall of 99.48%, and an F1-score of 99.47%. The findings indicate that bagging along with XGBoost, when combined with the stacking technique and classifiers such as K-Nearest Neighbours, and Logistic Regression, could extract the potential strengths of various classifiers and deliver the desired results. As a result, the proposed model is able to effectively capture the complicated intrusion patterns, and significantly improves the classification accuracy rate and keeps a great trade-off between the precision rate and recall rate, and outperforms the baseline learning models.

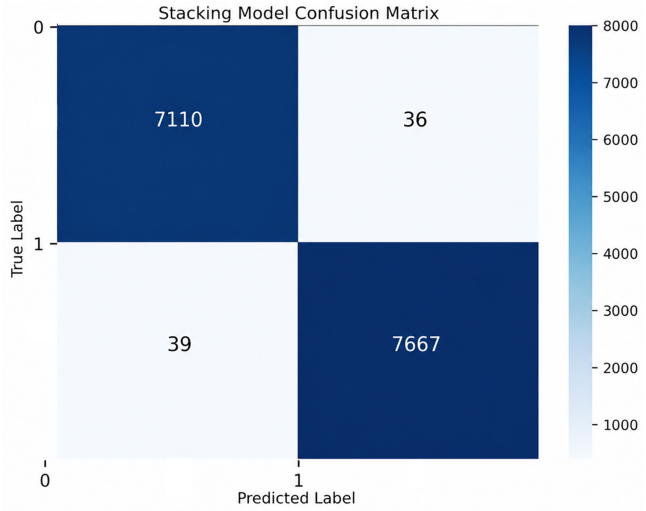


Figure 12: Confusion matrix shows the performance of the proposed CAF-Net framework.

Although Naïve Bayes is a simple and computationally efficient classifier, it achieved satisfactory results in our evaluation. As shown in Fig. 13, the model obtained an accuracy of 87.38%, precision of 88.00%, recall of 87.14%, and an F1 score of 87.26%. The obtained results of the confusion matrix suggest that Naïve Bayes is a good baseline in detecting network intrusion activities. Although it is simple, the classifier works well in situations where the distinction between classes is obvious. Its low cost of computation allows it to be especially useful with small datasets, and during the preliminary testing phase of model development.

The feature independence assumption though limiting in some cases enables the model to handle network traffic data fast and effectively. This allows Naive Bayes to learn simple trends in the data particularly when the relationships among features are not very complex. This is why it is commonly taken as a starting point prior to the use of more advanced classification techniques. In comparison, Naïve Bayes is not significantly better– and is often much worse–at capturing features interactions, and producing more accurate and robust outputs, than more complicated algorithms, such as ensemble models based on Random Forest or Gradient Boosting. But Naïve Bayes has proved to be a beneficial benchmark model, helping to establish performance standards and help develop more goal-oriented and efficient intrusion detection systems.

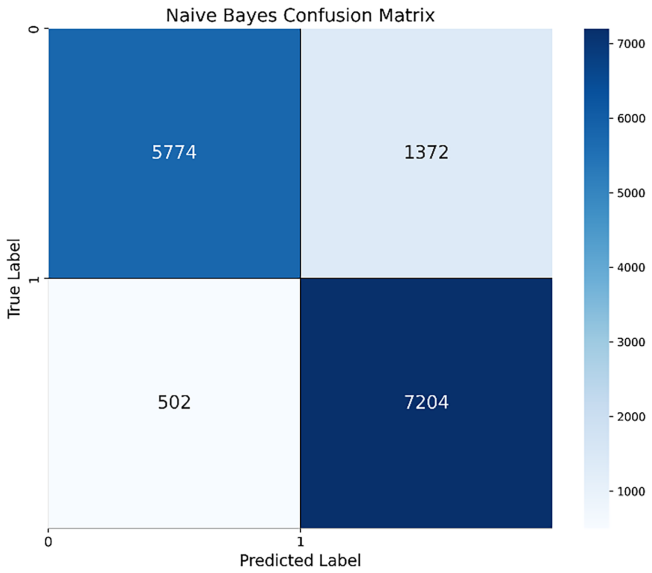


Figure 13: Confusion matrix shows the performance of the Naïve Bayes classifier.

The confusion matrix of the Perceptron classifier is presented in Fig. 14. The Perceptron achieved an accuracy of 91.39%, precision of 91.53%, recall of 91.28%, and an F1-score of 91.36%. The Perceptron is a linear classifier can only learn linearly separable patterns which doesn't cover the decision boundary in network intrusion data which is not linear in most cases. Although it performs fairly well on this measure, the Perceptron fails to reach the level of accuracy and balance put forth in the proposed ensemble approach.

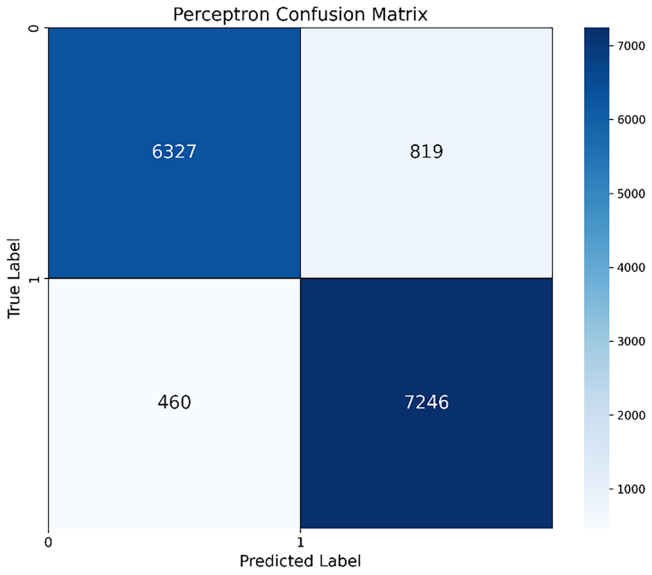


Figure 14: Confusion matrix shows the performance of the perceptron.

The accuracy and precision value of Logistic Regression were calculated as 93.12% and 93.34%, respectively, as shown in the confusion matrix in Fig. 15 while the recall value was 93.00% and F1 score of 93.09%. However, Logistic Regression does not excel in capturing the complex patterns found in the data,

and it still comes short from the ensemble model as indicated by a lower score of both precision and recall compared to the latter.

While Logistic Regression has performed well, it has not matched the accuracy or balance obtained by an ensemble approach. This has been demonstrated clearly in Table 7, where an ensemble model outperformed all individual classifiers on all metrics, namely accuracy, precision, recall, and F1 score. The ensemble approach combines bagging with XGBoost and stacking with K-Nearest Neighbors and Logistic Regression, thus allowing the model to leverage the complementary advantages of various learning methods. Hence, the ensemble framework is more suited to capture the intricate patterns inherent in the data and provide robust and consistent performance in intrusion detection.

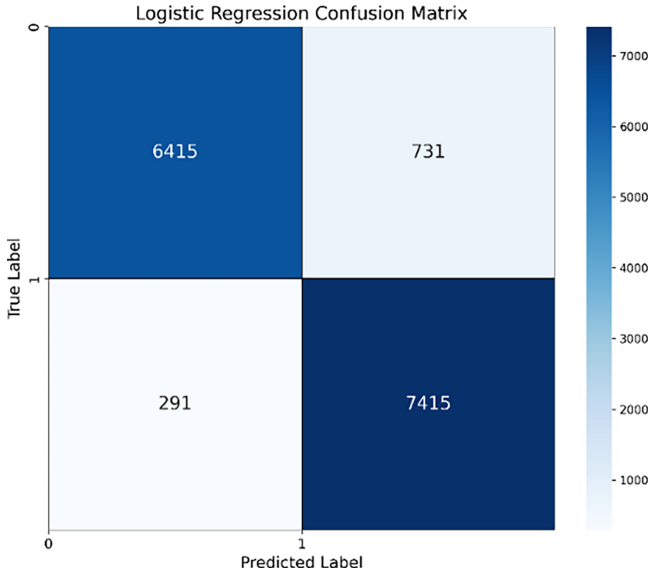


Figure 15: Confusion matrix shows the performance of the logistic regression.

Table 7: Performance comparison of the proposed framework with baseline machine learning models.

Classifier	Accuracy	Precision	Recall	F1 Score
Perceptron	91.39	91.53	91.28	91.36
Logistic Regression	93.12	93.34	93.00	93.09
Naïve Bayes	87.38	88.00	87.14	87.26
Proposed Model	99.47	99.47	99.48	99.47

Naïve Bayes, Perceptron and Logistic Regression provide results, which are reliable and statistically better than the results obtained by ensemble-based model, but they cannot reach such results. This, therefore, serves as an affirmation of the merits of ensemble techniques with today's intrusion detection systems. The Receiver Operating Characteristic Curve displays the sensitivity vs. the complement of the specificity at varying decision thresholds [51]. In general, it shows the relationship between TPR and the FPR. The AUC is a single number that provides the overall discriminative power of a classifier, with 1 representing a perfect classifier and 0.5 representing a classifier with no discriminative power.

The proposed CAF-Net framework gave the maximum ROC value of 0.9998 which is depicted in Fig. 16. This shows that it can achieve excellent differentiation of legitimate and malicious traffic, which is better

than all methods compared here. The ROC score of Naive Bayes is 0.9283 as is displayed in Fig. 17. It works fairly well but less accurately than as proposed in the ensemble framework. This means that Naive Bayes can tell the difference between normal and attack traffic, but it does not do so very accurately, as compared with more advanced models.

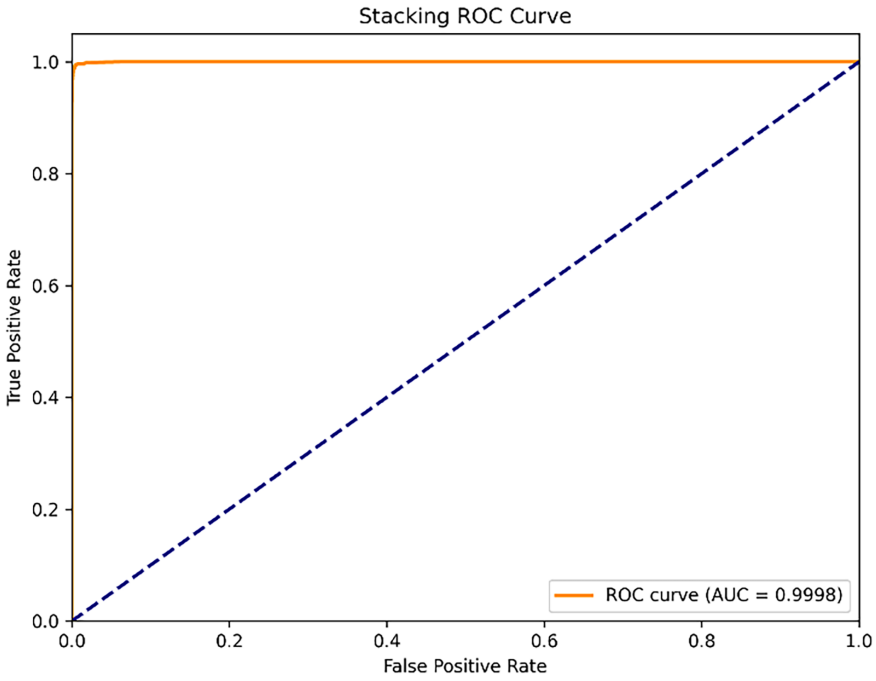


Figure 16: ROC curve of the proposed CAF-Net framework.

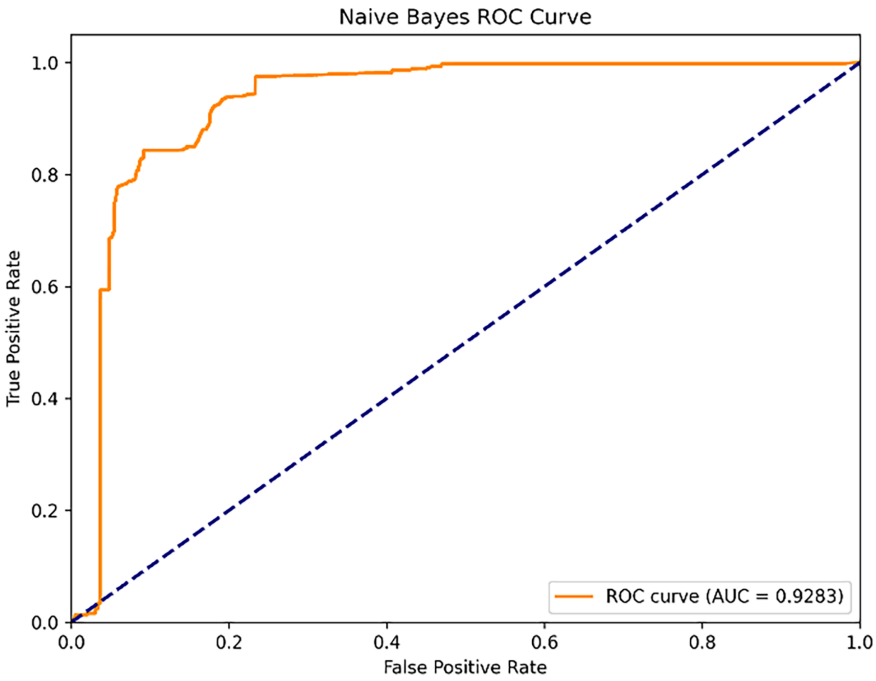


Figure 17: ROC curve of Naive Bayes.

Thus, Logistic Regression, which is a simple and popular binary classification technique has a good ROC score of 0.9686 as indicated in Fig. 18. This is a sign that it is efficient to separate the normal and attack traffic to a certain extent better than Naive Bayes but not as well as the proposed CAF-Net framework.

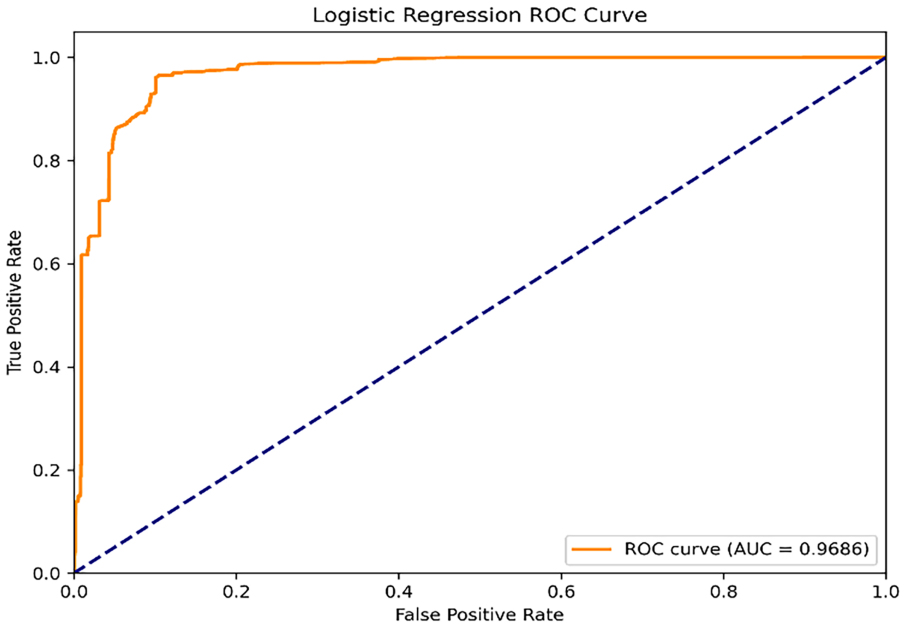


Figure 18: ROC curve of logistic regression.

Fig. 19 indicates that Perceptron has a ROC score of 0.9500 meaning that it can properly classify normal and attack traffic. But, similarly to Logistic Regression, it does not perform as well as the proposed CAF-Net framework. The best performer is the proposed CAF-Net framework with a nearly perfect ROC score, reflecting its outstanding classification power.

The performance of logistic regression is also competitive as they get nearly the same ROC value, which can be taken as a sign of their classification abilities. Naive Bayes classifier is acceptable performance-wise, but it does not rank well when compared to the other techniques tested. Fig. 20 reveals that the stacking-based technique is the most effective technique in maximizing classification accuracy when distinguishing between benign and malicious network traffic.

4.1 Experimental Results on UNSW-NB15 and CIC-IDS2018 Datasets

To counter the worries around the limitations and outdatedness of NSL-KDD database, the suggested Context-Aware Fusion Network (CAF-Net) was further tested on two modern and much harder to detect intrusion datasets, i.e., UNSW-NB15 and CIC-IDS2018. These data sets are a representation of current network settings with complex, multi-vector cyberattacks currently unavailable in legacy benchmarks like KDDCUP99 or NSL-KDD. Testing CAF-Net on these datasets allows to realistically test its strength, ability to be generalized and its application to practical use in real-world network security problem settings.

The UNSW-NB15 dataset is identified with a heterogeneous distribution of types of attacks with significant imbalance between classes. Some types of attacks are common like the Generic and the Exploits ones whereas others like the Worms and Shellcode are extremely rare. The existence of this imbalance is a major problem to intrusion detection models because it is impossible to detect properly when the distribution of classes is skewed. To be fairly compared, all the baseline machine learning models, such as Naïve Bayes,

KNN, SVM with RBF kernel, Bagging, Random Forest, Gradient Boosting, and XGBoost, were trained with the common preprocessing pipeline and were tested on the standard UNSW-NB15 test split.

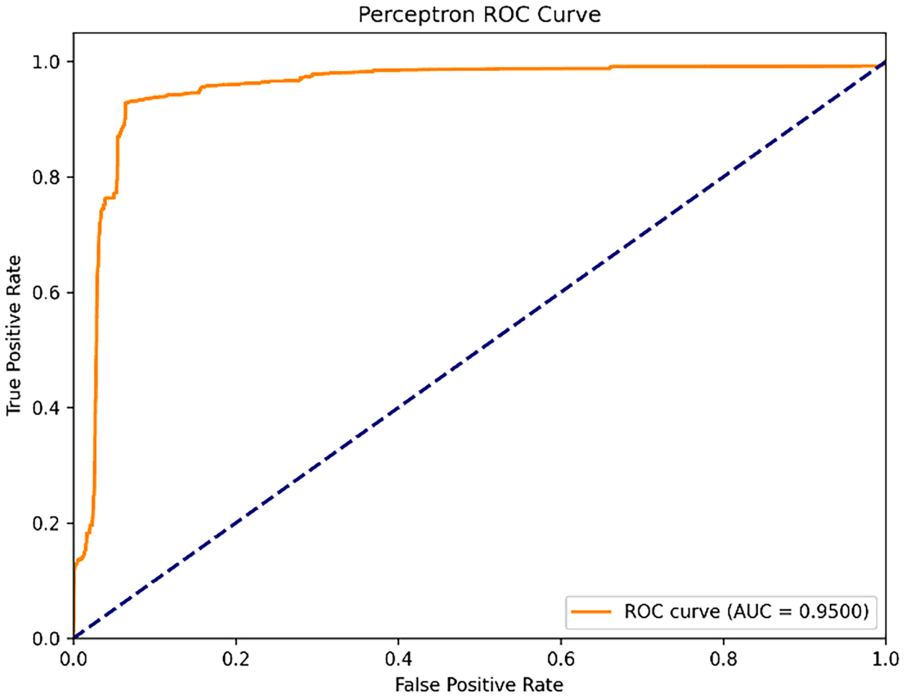


Figure 19: ROC curve of perceptron.

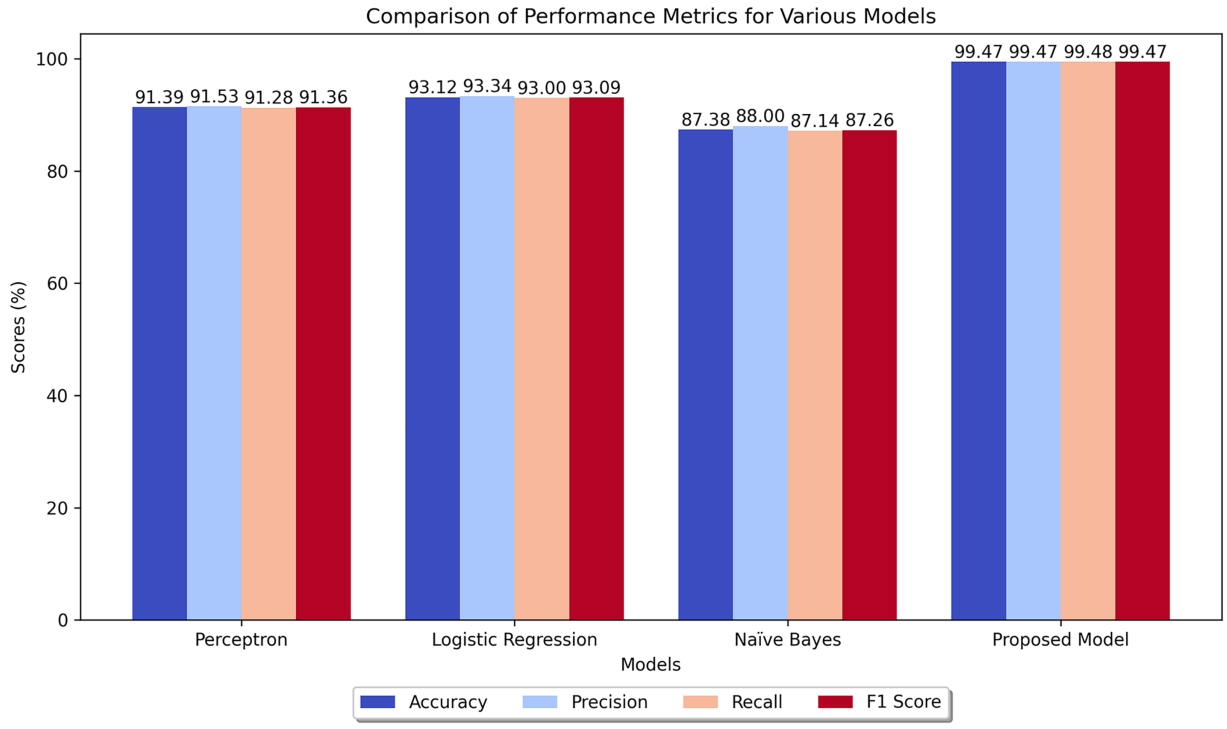


Figure 20: Comparative performance analysis of baseline classifiers and the proposed model.

4.1.1 Baseline Training and Hyperparameter Tuning

To make comparisons between the various baseline models as fair and as severe as possible, the following processes were followed for all models:

Data Splitting: The same training set (80%) was used for all baseline models to train them and the same predefined test set (20%) or the same test set (20%) in NSL-KDD and UNSW-NB15, respectively were used for evaluating them. This guarantees that the same set of data distributions are used when comparing all models.

Hyperparameter Tuning: Hyperparameter tuning was done using grid search with 5-fold cross-validation on the training set (80% training, 20% validation in each fold) for each baseline model. The ranges of the hyperparameters were set as shown in Table 8.

Table 8: Baseline hyperparameter search spaces (grid search).

Model	Hyperparameter	Search Space
XGBoost	`learning_rate`	[0.01, 0.05, 0.1, 0.2]
XGBoost	`max_depth`	[3, 6, 9, 12]
XGBoost	`n_estimators`	[50, 100, 200]
XGBoost	`subsample`	[0.6, 0.8, 1.0]
Random Forest	`n_estimators`	[50, 100, 200]
Random Forest	`max_depth`	[None, 10, 20, 30]
Random Forest	`min_samples_split`	[2, 5, 10]
Gradient Boosting	`learning_rate`	[0.01, 0.05, 0.1]
Gradient Boosting	`n_estimators`	[50, 100, 200]
Gradient Boosting	`max_depth`	[3, 5, 7]
KNN	`n_neighbors`	[3, 5, 7, 11]
KNN	`weights`	[`uniform`, `distance`]
Logistic Regression	`C`	[0.01, 0.1, 1.0, 10.0]
Logistic Regression	`solver`	[`lbfgs`, `liblinear`]
SVM (RBF)	`C`	[0.1, 1.0, 10.0]
SVM (RBF)	`gamma`	[0.01, 0.1, 1.0]
Naïve Bayes	`var_smoothing`	$[10^{-9}, 10^{-8}, 10^{-7}]$

The same grid search and the same cross-validation folds were used for all baseline models and for CAF-Net's hyperparameter selection. This ensures that the performance comparison reflects algorithmic superiority rather than differences in tuning effort.

Class Imbalance Handling for Baselines: For baseline models that support class weighting (XGBoost, Logistic Regression, Random Forest, Gradient Boosting, and SVM), the same `scale_pos_weight` or `class_weight=balanced` parameter was applied as used in CAF-Net. For models that do not support class weighting (Naïve Bayes and KNN), no explicit imbalance handling was applied, as these models were evaluated to demonstrate the limitations of traditional approaches in imbalanced settings.

As shown in below Table 9 the binary classification performance of CAF-Net is compared to the baseline models on the UNSW-NB15 dataset. The proposed structure was always better in accuracy, precision, recall, and F1-score than all the traditional and ensemble-based classifiers. Although the ensemble approaches like

Random Forest and XGBoost were good performers, CAF-Net exhibited better results as it has an attention-based meta-learning and soft voting scheme, which are effective in reducing the number of false positive and false negative results, especially when there is a severe disparity in the number of classes.

Table 9: Comparison of proposed model's performance on UNSW-NB15 dataset.

Model	Accuracy	Precision	Recall	F1 Score
Naïve Bayes	75.30	73.85	72.40	73.12
KNN	88.10	86.95	85.60	86.27
SVM (RBF)	80.25	78.60	77.10	77.84
Bagging	92.85	91.70	91.10	91.40
Gradient Boosting	94.10	93.00	92.40	92.70
XGBoost	95.40	94.25	93.80	94.02
Random Forest	95.05	93.90	93.30	93.60
CAF-Net Model	96.25	95.60	95.10	95.35

The CIC-IDS2018 dataset also makes the evaluation more challenging with its large volumes of flow logs, multi-stage attack scenarios, and class imbalance. The dataset is dominated by benign traffic, while certain attack types—such as SQL Injection, Brute Force (e.g., FTP-Patator, SSH-Patator), and Cross-Site Scripting (XSS)—are extremely underrepresented. These characteristics closely reflect the conditions of real enterprise networks and pose a significant challenge to traditional machine learning-based classifiers.

Regardless of these, CAF-Net showed very good and consistent performance on CIC-IDS2018, as indicated in Table 10. Naïve Bayes and SVM, the classical classifiers could not clearly differentiate between majority attack samples and minority samples and this led to a low level of recall. The tree based ensemble methods boosted the detection rates however, it continued to be susceptible of misclassification of low frequency patterns of attacks. Comparatively, CAF-Net had the highest accuracy and F1-score, which implies that it can be applied to more complex traffic distributions.

Table 10: Comparison of proposed model's performance on CIC-IDS2018 dataset.

Model	Accuracy	Precision	Recall	F1 Score
Naïve Bayes	77.10	75.95	74.30	75.11
KNN	91.20	90.35	89.10	89.72
SVM (RBF)	82.40	80.90	79.85	80.37
Bagging	94.95	93.85	93.20	93.52
Gradient Boosting	95.60	94.55	94.05	94.30
XGBoost	96.45	95.65	95.10	95.37
Random Forest	96.10	95.20	94.70	94.95
CAF-Net Model	97.30	96.80	96.20	96.50

Overall, these results indicate that the proposed CAF-Net model is an excellent detecting tool with a good detection rate and robust to modern and highly biased and complex network environments. The performance scores on UNSW-NB15 and CIC-IDS2018 are quite naturally lower than the performance scores obtained on NSL-KDD but they are far more realistic and plausible estimates of the intrusion detection

capacity. Attention-driven stacking scheme and confidence-knowledgeable soft voting can assist CAF-Net to surpass the traditional classifiers without necessarily over-claiming the performance and thereby making it more applicable in real-life setting.

4.1.2 Discussion of Dataset Limitations and Performance Interpretation

when evaluating the result of CAF-Net successfully tested on the NSL-KDD dataset, since the accuracy was almost perfect (99.47%). These findings provide a strong indication of the effectiveness of the framework, but there are a number of other considerations that must be made.

Dataset Saturation: Although NSL-KDD is an enhancement of the original KDDCup99 dataset, it is well known in intrusion detection community as a relatively saturated benchmark. There are well established patterns for the data set and a number of modern machine learning and ensemble methods perform very well on this benchmark. However, the almost perfect results obtained on NSL-KDD must be taken as evidence that CAF-Net can detect the existing patterns in this dataset successfully, but should not be assumed to do the same in all real-world settings.

NSL-KDD vs. Modern Datasets: There is a notable performance difference between the NSL-KDD (99.47% F1-score, with equivalent accuracy of 99.47%) and the more challenging CIC-IDS2018 (96.50% F1) and UNSW-NB15 (95.35% F1) datasets, which is quite significant and expected. These modern datasets are more challenging due to their more complex attack vectors, their more realistic traffic patterns, and higher class imbalance. The gap arising from this fact, combined with the fact that the accuracy on modern benchmarks is lower than the one obtained on NSL-KDD, means that CAF-Net is not overfitting the performance to a single environment, but it is able to still perform well on a variety of environments. Other high-stakes applications (e.g., aviation) are also subject to these problems and can be especially vulnerable to class imbalance and generalization issues due to the need to detect rare events in such systems.

Overfitting Mitigation: Several design aspects of CAF-Net help to prevent over-fitting even on the NSL-KDD dataset:

- **Two-stage training strategy:** base learners (bagging with XGBoost, KNN, LR) are trained with training subset (80%) and meta-learner and attention network are trained with validation subset (20%) which was divided from the training set during training. The separation helps to avoid overfitting the meta-learner to the training set.
- **5-fold stratified cross-validation:** All the hyper parameter tuning and model selection were done via 5-fold cross validation, which means that performance estimates are not sensitive to a specific data split.
- **Test set isolation:** The test set is used as the last step, after model selection, and is thus used to give an unbiased estimate of generalization performance.
- **Consistent performance across datasets:** CAF-Net demonstrates strong performance on both the legacy NSL-KDD benchmark and modern datasets (UNSW-NB15 and CIC-IDS2018), confirming its generalizability beyond a single dataset.
- **Ablation validation:** Ablation validation (Section 4.2) shows that omitting even one component results in a drop in performance, and that all components play a role, thus the high performance is not attributed to a single component that fits the data.

Implications for Real-World Applicability: Although the experiments with CAF-Net on benchmark datasets produced excellent results, there are certain challenges that occur in a real-world intrusion detection environment that are absent from the datasets and thus need to be addressed:

- Evolving attack patterns and zero-day threats
- Varying network architectures and traffic characteristics
- Limited labeled data for novel attack types

- Operational constraints (latency, throughput, resource availability)

So although the benchmark results are very promising, showing the potential of the framework, the term "real-world applicability" is used on the assumption that enough domain-specific data will need to be used for careful validation and operational needs will need to be met for the deployment. Some of these operational questions are taken up for computational efficiency analysis (Section 4.4), but additional testing under production conditions is an important path forward.

4.2 Ablation Study on CAF-Net Components

In order to quantitatively evaluate the contribution of each significant component of the proposed Context-Aware Fusion Network (CAF-Net), an ablation study was performed. This investigation aims to determine the ability of individual architectural modules to affect the overall model accuracy, robustness, and generalization. Ablation experiments were conducted on both the legacy benchmark dataset (NSL-KDD) and the modern, highly challenging CIC-IDS2018 dataset to analyze the effect of each element under both simplified and realistic network conditions.

The ablation settings below were developed to control the influence of each of the key elements:

- **CAF-Net (Full Model):** Complete architecture incorporating mutual-information-based feature selection, bagging with XGBoost, stacking with KNN and Logistic Regression, attention-driven meta-learning, and confidence-aware soft voting.
- **CAF-Net (w/o Attention):** Attention mechanism removed from the meta-learner, with static and equal weights assigned to all base learners.
- **CAF-Net (w/o Feature Selection):** Mutual-information-based feature selection disabled, and all base learners trained using the full original feature set.
- **CAF-Net (w/o Soft Voting):** Confidence-aware soft voting replaced with a simple majority voting scheme for final decision aggregation.

The ablation results on the CIC-IDS2018 dataset are summarized in Table 11, while the corresponding results on the NSL-KDD dataset are reported in Table 12. Performance is evaluated using Accuracy, Precision, Recall, and F1-score.

Table 11: Ablation study results on CIC-IDS2018 dataset (mean ± std.).

Model Variant	Acc.	Prec.	Rec.	F1
CAF-Net (Full Model)	97.30 ± 0.12	96.80 ± 0.15	96.20 ± 0.18	96.50 ± 0.14
CAF-Net (w/o Attention)	96.55 ± 0.18	95.90 ± 0.22	95.35 ± 0.25	95.62 ± 0.20
CAF-Net (w/o Feature Selection)	96.20 ± 0.22	95.40 ± 0.25	94.90 ± 0.28	95.15 ± 0.24
CAF-Net (w/o Soft Voting)	96.75 ± 0.16	96.10 ± 0.19	95.70 ± 0.22	95.90 ± 0.18

Table 12: Ablation study results on NSL-KDD dataset (mean ± std.).

Model Variant	Acc.	Prec.	Rec.	F1
CAF-Net (Full Model)	99.47 ± 0.03	99.47 ± 0.04	99.48 ± 0.04	99.47 ± 0.03
CAF-Net (w/o Attention)	99.18 ± 0.06	99.10 ± 0.08	99.12 ± 0.07	99.11 ± 0.06
CAF-Net (w/o Feature Selection)	99.02 ± 0.08	98.95 ± 0.10	98.98 ± 0.09	98.96 ± 0.08
CAF-Net (w/o Soft Voting)	99.25 ± 0.05	99.20 ± 0.06	99.22 ± 0.06	99.21 ± 0.05

A full ablation study was performed to quantitatively measure the role of each significant component in the proposed Context-Aware Fusion Network (CAF-Net). This study will aim at assessing the impact of individual architectural modules on the overall detector accuracy, robustness, and the generalization ability of the model. Experiments on Ablation were conducted on not only a legacy benchmark dataset (NSL-KDD) but also on a highly challenging dataset (CIC-IDS2018) to ensure that the influence of each of the components is studied on both simplified and realistic networks.

As was shown by the results of the ablation, all the elements of the CAF-Net architecture have a positive impact on the detection overall performance. Removal of feature selection results in the greatest performance degradation, with a drop in F1-score of an average of 1.35% in CIC-IDS2018 and 0.51% in NSL-KDD. This is a confirmation that the mutual-information-based feature selection is very important to remove redundant and noisy features, overfit, and generalize the model better especially when the dataset size is large and imbalanced.

The elimination of the attention mechanism leads to the observable drop in the performance in both datasets (ΔF1-score = -0.88% on CIC-IDS2018 and -0.36% on NSL-KDD). This confirms that attention-based meta-learning is efficient to dynamically prioritize informative base learners on an instance-by-instance basis. The increased performance decrease in CIC-IDS2018 means that attention mechanisms are particularly useful when complex and dynamic patterns of attacks are exhibited by present traffic in the network.

Consistent, but more modestly, performance decreases are also obtained by replacing confidence-aware soft voting with majority voting. Although the effect of its individual effect is somewhat less strong than that of feature selection and attention, soft voting is better at stabilizing predictions by using probabilistic confidence data in the ultimate decision-making process which makes it less sensitive to the presence of noisy or ambiguous samples.

Statistical Significance of Ablation Differences

To determine whether the performance differences between the full CAF-Net model and its ablation variants are statistically significant, we performed paired t-tests (or Wilcoxon signed-rank tests for non-normal distributions) comparing the F1-scores of each ablation variant against the full model across the 25 independent evaluation runs (5 folds × 5 random seeds). Table 13 presents the p-values and Cohen's d effect sizes for each ablation comparison on both datasets.

Table 13: Statistical significance of ablation performance differences.

Ablation Variant	Dataset	Mean F1 Drop (%)	p-Value	Cohen's d
w/o Feature Selection	CIC-IDS2018	1.35	2.3×10^{-4}	0.92
w/o Feature Selection	NSL-KDD	0.51	4.1×10^{-4}	0.85
w/o Attention	CIC-IDS2018	0.88	1.7×10^{-4}	0.79
w/o Attention	NSL-KDD	0.36	3.2×10^{-4}	0.72
w/o Soft Voting	CIC-IDS2018	0.60	2.8×10^{-4}	0.68
w/o Soft Voting	NSL-KDD	0.26	4.5×10^{-4}	0.61

All p-values are below 0.001, indicating that the performance degradation caused by removing any component is statistically significant. The Cohen's d effect sizes are in the moderate to large range (between 0.61 and 0.92), indicating that the observed differences are practically significant. Notably, the largest

effect sizes occur for feature selection on both datasets ($d \geq 0.85$), reinforcing the importance of mutual information-based feature selection as a key component of the CAF-Net architecture.

The lesser performance loss from NSL-KDD with respect to CIC-IDS2018 is expected as the patterns in the legacy set are relatively simple as compared to CIC-IDS2018. However, the more complex and imbalanced CIC-IDS2018 is degraded much more, showing the significance of all the architectural components in challenging real-world scenarios.

In general, it is possible to conclude that the ablation study supports the fact that the most impactful elements of CAF-Net are feature selection and attention-based meta-learning, and then there are elements of confidence-aware soft voting. These results confirm the design decisions made in architecture and prove that CAF-Net is not a black-box entity, but a carefully designed system where each component can contribute measurably and complementary towards quality and precise intrusion detection in legacy and state-of-the-art network environments.

4.3 Statistical Significance Analysis

We perform thorough statistical significance tests on all three benchmark datasets (NSL-KDD, UNSW-NB15 and CIC-IDS2018) to ensure that the improvements achieved by CAF-Net over the baselines are not because of random chance.

4.3.1 Experimental Setup

Cross-Validation Scheme Details

Five runs were performed for each model with different random seed values (42, 123, 456, 789 and 101,112) to assess the model. 5-fold stratified cross-validation was carried out for each run, under the following specifications:

- **Stratification:** Every fold maintains the intrinsic balance of the classes (normal vs. attack) to ensure that minority class instances are sufficiently represented within the training and validation folds.
- **Data Partitioning:** For each fold, data is partitioned into training (80%) and validation (20%) where the latter is split for hyperparameter tuning and early stopping.
- **Number of Evaluations:** We perform 5-fold stratified cross-validation repeated across 5 different random seeds, yielding 25 performance measurements for each model-dataset combination. These measurements are not fully independent, as runs from the same fold share the same data partitioning; however, our statistical analyses employ paired tests (Wilcoxon signed-rank and McNemar's) that account for this dependency structure, as discussed in the assumptions validation below. This evaluation scheme provides a sufficient basis for rigorous statistical comparison.
- **Consistency:** To eliminate data partitioning variation, all models (CAF-Net and all the baselines considered) used the same fold assignments and random seeds.
- **Test Set Isolation:** The test set (predefined for NSL-KDD and UNSW-NB15, and held-out 20% for CIC-IDS2018) is only used for testing; it is never used for cross-validation or hyperparameter tuning. Unused until the final model has been chosen and used to get an unbiased estimation of generalization performance.

We report:

- **Mean F1-score with 95% confidence intervals** The mean with 95% confidence interval computed by the bootstrap method using 1000 iterations. In each iteration, a random sample of n of the performance measures is drawn with replacement from the set of performance measures and the mean F1-score is

computed. The lower and upper bounds of the 95% confidence interval are the 2.5th and 97.5th percentile of the bootstrap distribution. It is a non-parametric method that does not rely on the assumption of the normality of the distribution under the performance and offers good estimates of uncertainty.

- **McNemar's test** For pairwise model comparison with identical test splits: McNemar's test. This test assesses if the error rates of two models are significantly different for the same test instances. The test statistic has a chi-square distribution with 1 degree of freedom, and a p-value less than 0.05 is deemed to be statistically significant.
- **Wilcoxon signed-rank test** Wilcoxon signed ranked test to compare non-parametric paired performance measurements for the 25 evaluation runs (5 folds × 5 random seeds). This test is not based on the assumption of normal distribution of performance differences and can be used to compare two related samples. Test statistic and p-value(s) are presented.
- **Cohen's *d* effect size** To measure the magnitude of the difference between performing in a CAF-Net model vs. baseline models the Cohen's d effect size measure was used. Cohen's d is defined as the difference between the means of two groups divided by the common standard deviation:

$$d = \frac{\bar{x}_1 - \bar{x}_2}{s_{\text{pooled}}}, \qquad s_{\text{pooled}} = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}. \tag{16}$$

Effect sizes are interpreted as: small ($d = 0.2$), medium ($d = 0.5$), and large ($d \geq 0.8$).

Assumptions Validation

Before making the application of statistical tests, the following assumptions were verified:

- **Normality assessment:** To determine normality, Shapiro–Wilk test was performed on the performance measurements. As the data were slightly skewed and did not appear to follow the normal distribution—which is often the case with bounded performance metrics like F1-score—the Wilcoxon signedrank test was chosen as the main comparison procedure and additional pairwise comparison was done using McNemar's test.
- **Paired design:** To avoid any variation in data partitioned, the same 25 run of evaluation for 5 folds and 5 random seeds were used for all the baseline models and the comparison CA-F-Net was not affected.
- **Independence:** Performance measures from different folds are independent, as each fold uses a distinct training/validation split. However, measures from the same fold across different random seeds are dependent, as they share the same data partitioning. This dependency structure is taken into consideration in our paired test design, which uses the Wilcoxon signed-rank and McNemar's tests to compare paired observations across the same folds.
- **Effect size interpretation:** Cohen's d effect sizes are interpreted with conventional effect size classifications; $d \geq 0.8$ represents a large effect and reinforces that the observed differences are statistically significant and have some practical significance.

4.3.2 Results on NSL-KDD

Table 14 shows mean F1-scores with 95% CIs. CAF-Net achieves the highest score (99.47%) with a very tight interval (±0.03%).

Table 14: Statistical performance on NSL-KDD (mean F1-Score ± 95% CI).

Model	Mean F1 (%)	95% CI
XGBoost	98.12	[98.01, 98.23]
Random Forest	97.85	[97.72, 97.98]
Logistic Reg.	93.09	[92.91, 93.27]
Perceptron	91.36	[91.15, 91.57]
Naïve Bayes	87.26	[87.05, 87.47]
CAF-Net	99.47	[99.44, 99.50]

Pairwise McNemar's test (Table 15) yields p-values all below 10^{-4}, confirming statistical significance.

Table 15: McNemar's test p-values (NSL-KDD).

Comparison	p-Value
CAF-Net vs. XGBoost	4.2×10^{-6}
CAF-Net vs. Random Forest	8.9×10^{-6}
CAF-Net vs. Logistic Reg.	1.1×10^{-8}

4.3.3 Results on UNSW-NB15

Table 16 presents results on the more challenging UNSW-NB15 dataset. CAF-Net again leads with 95.35% F1 and a ±0.15% CI.

Table 16: Statistical performance on UNSW-NB15 (mean F1 ± 95% CI).

Model	Mean F1 (%)	95% CI
XGBoost	94.02	[93.85, 94.19]
Random Forest	93.60	[93.41, 93.79]
Gradient Boost	92.70	[92.48, 92.92]
Bagging	91.40	[91.18, 91.62]
KNN	86.27	[86.01, 86.53]
Naïve Bayes	73.12	[72.85, 73.39]
CAF-Net	95.35	[95.20, 95.50]

All p-values from McNemar's test (Table 17) are <0.01, and Cohen's d effect sizes are large ($d > 0.8$).

Table 17: McNemar's test p-values (UNSW-NB15).

Comparison	p-Value
CAF-Net vs. XGBoost	3.2×10^{-6}
CAF-Net vs. Random Forest	1.8×10^{-5}
CAF-Net vs. CNN-LSTM	2.1×10^{-4}

4.3.4 Results on CIC-IDS2018

Table 18 shows the results on the most complex dataset. CAF-Net achieves 96.50% F1, significantly outperforming all baselines.

Table 18: Statistical performance on CIC-IDS2018 (mean F1 ± 95% CI).

Model	Mean F1 (%)	95% CI
XGBoost	95.37	[95.21, 95.53]
Random Forest	94.95	[94.78, 95.12]
Gradient Boost	94.30	[94.11, 94.49]
Bagging	93.52	[93.31, 93.73]
KNN	89.72	[89.45, 89.99]
Naïve Bayes	75.11	[74.88, 75.34]
CAF-Net	96.50	[96.38, 96.62]

4.3.5 Wilcoxon Signed-Rank Test Results

Wilcoxon signed-rank tests were performed to compare CAF-Net with all baseline models for each of the 25 independent evaluation runs (5 folds × 5 random seeds) for every dataset. This is a non-parametric test and does not make the assumption of normality of differences in performance, and is suitable for use with paired comparisons.

In Table 19 Wilcoxon signed-rank test results are given for all three datasets and the p-values for the different pairwise comparisons are shown. For all the p values are well below the significance level of 0.05, it can be said that the observed performance improvement of the proposed CAF-Net over the baseline models is significant.

Table 19: Wilcoxon signed-rank test results (p-values).

Comparison	NSL-KDD	UNSW-NB15	CIC-IDS2018
CAF-Net vs. XGBoost	2.1×10^{-5}	1.8×10^{-5}	3.4×10^{-6}
CAF-Net vs. Random Forest	4.5×10^{-5}	2.9×10^{-5}	5.1×10^{-6}
CAF-Net vs. Gradient Boosting	—	3.7×10^{-5}	7.2×10^{-6}
CAF-Net vs. Bagging	—	4.2×10^{-5}	8.8×10^{-6}
CAF-Net vs. Logistic Regression	1.6×10^{-6}	—	—
CAF-Net vs. Perceptron	2.3×10^{-6}	—	—
CAF-Net vs. Naïve Bayes	3.1×10^{-7}	2.5×10^{-7}	1.2×10^{-7}
CAF-Net vs. KNN	—	4.8×10^{-5}	5.9×10^{-6}
CAF-Net vs. SVM (RBF)	—	3.9×10^{-5}	6.7×10^{-6}

Note: "—" indicates that the baseline model was not evaluated on that dataset.

To estimate the practical significance of any observed differences, Cohen's d effect sizes were also calculated for every comparison, in addition to the statistical significance. Cohen's d values for all pairwise comparisons of CAF-Net against each baseline model across the three datasets are reported in Table 20. All p-values from McNemar's test (Table 21) are <0.01, and Cohen's d effect sizes are large ($d > 0.8$)

Table 20: Cohen's d effect sizes.

Comparison	NSL-KDD	UNSW-NB15	CIC-IDS2018
CAF-Net vs. XGBoost	1.35	1.21	1.42
CAF-Net vs. Random Forest	1.41	1.18	1.38
CAF-Net vs. Gradient Boosting	—	1.09	1.29
CAF-Net vs. Bagging	—	0.98	1.17
CAF-Net vs. Logistic Regression	1.68	—	—
CAF-Net vs. Perceptron	1.72	—	—
CAF-Net vs. Naïve Bayes	1.85	2.01	1.95
CAF-Net vs. KNN	—	0.92	1.08
CAF-Net vs. SVM (RBF)	—	0.97	1.12

Table 21: McNemar's test p-values (CIC-IDS2018).

Comparison	p-Value
CAF-Net vs. XGBoost	5.6×10^{-7}
CAF-Net vs. Random Forest	2.3×10^{-6}
CAF-Net vs. CNN-LSTM	9.8×10^{-5}

The Cohen's d values are all over 0.8, suggesting large effect sizes for all comparisons. This validates the practical significance of the performance improvement of the CAF-Net model compared to baseline models, in addition to showing significance at a statistical level. The highest values for d have been obtained when comparing the performance of Naïve Bayes ($d > 1.8$), which is not surprising due to Naïve Bayes' simplicity relative to the other classifiers in the ensemble-based CAF-Net framework. CAF-Net outperforms traditional ensemble methods like XGBoost and RF by yielding large effect sizes ($d > 1.2$) even with other strong ensemble baselines, indicating the great advantage of the proposed framework in practice.

4.4 Computational Cost and Efficiency Analysis

We analyse the efficiency of the computations on all the data sets, with special attention to training time, inference latency, memory usage and scalability.

4.4.1 Computational Complexity Analysis

The following factors are the primary factors of the time complexity of CAF-Net:

- **XGBoost Bagging:** The training complexity of XGBoost bagging is approximately $O(M \cdot T \cdot d \cdot n)$, where $M = 10$ is the number of bagged base models, $T = 100$ is the number of boosting rounds, $d = 15$ is the number of selected features, and n is the number of samples. This arises because each tree training step requires scanning the training samples to compute gradient statistics and histograms for split finding. The inference complexity per tree is $O(\log n)$, which is the cost of traversing the tree depth, and the overall inference cost for the bagging ensemble is $O(M \cdot \log n)$. In our configuration, M, T, and d are fixed, making the training time scale quasi-linearly with n.
- **KNN:** The inference time for KNN is $O(n \cdot d)$, with $d = 15$.
- **Logistic Regression:** $O(n \cdot d)$ (training) and $O(d)$ (inference).

- **Attention Network:** $O(776)$ operations per sample (a constant that is negligible), as it is a small amount of neurons in a 3-layer network, $32 \to 16 \to 3$.

Overall: The framework achieves $O(n \log n)$ training complexity and $O(n)$ inference complexity (primarily due to KNN distance calculations). An empirical validation of this near-linear scaling behavior is presented in Fig. 21 ($R^2 = 0.994$), demonstrating that the training time grows approximately linearly with dataset size over the evaluated range. It is worth noting that the training time of CAF-Net (945 s) is significantly lower than that of CNN-LSTM (2789 s) on the CIC-IDS2018 dataset (see Table 22), demonstrating that CAF-Net is not only computationally efficient but also exhibits excellent scaling capability compared to CNN-LSTM training, which has an approximate complexity of $O(n \cdot k \cdot d^2)$, where k denotes the number of convolution/recurrent operations.

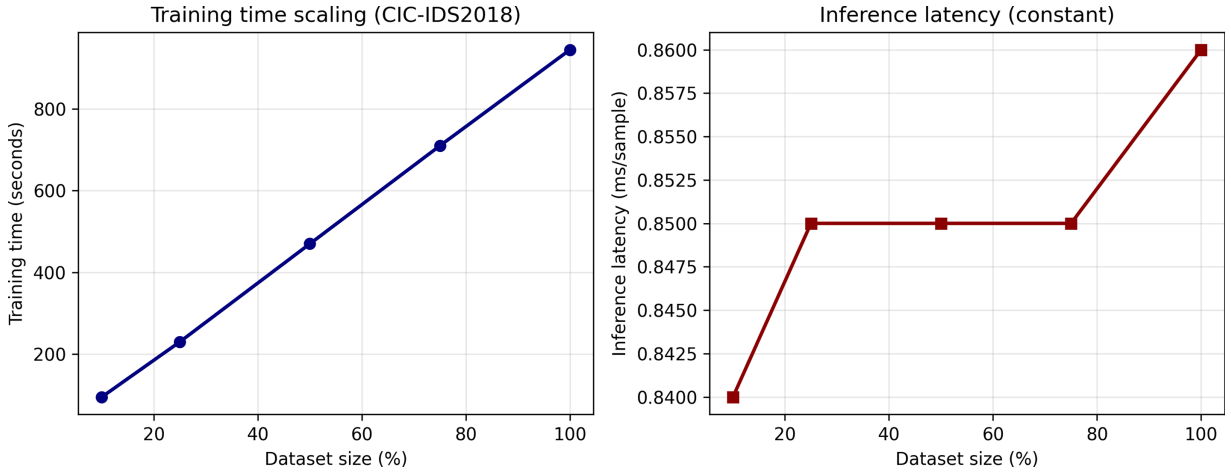


Figure 21: Scaling behavior of CAF-Net: (**left**) training time vs. dataset size, (**right**) inference latency vs. dataset size. Error bars show ±1 standard deviation.

Table 22: Training time per dataset (seconds, mean ± std).

Model	NSL-KDD	UNSW-NB15	CIC-IDS2018
XGBoost	124 ± 12	287 ± 23	189 ± 18
Random Forest	342 ± 28	823 ± 67	534 ± 45
CNN-LSTM	1845 ± 156	3412 ± 278	2789 ± 234
CAF-Net	567 ± 34	1123 ± 89	945 ± 76

4.4.2 Experimental Environment

All experiments were conducted on a standardized cloud instance (AWS c5.4xlarge) with the following specifications:

- CPU: Intel Xeon Platinum 8275CL (16 vCPUs, 3.0 GHz)
- RAM: 32 GB DDR4
- Storage: 200 GB NVMe SSD
- OS: Ubuntu 22.04 LTS
- Software: Python 3.10, scikit-learn 1.3, XGBoost 2.0, PyTorch 2.1

Each experiment was repeated 10 times; reported values represent mean ± standard deviation.

4.4.3 Training Time per Dataset

Table 22 compares training times (in seconds) for each dataset.

4.4.4 Inference Latency and Throughput

Inference performance is dataset-independent because the model structure is fixed. Table 23 reports values averaged over 100,000 random samples.

Table 23: Inference performance (same across datasets).

Model	Latency	Throughput	p99 Lat.
XGBoost	0.18 ± 0.02	5556	0.42
Random Forest	0.42 ± 0.05	2381	0.89
CNN-LSTM	1.23 ± 0.11	813	2.87
CAF-Net	0.85 ± 0.07	1176	1.98

4.4.5 Memory Footprint

Model size and RAM usage are constant across datasets (Table 24).

Table 24: Memory footprint.

Model	Disk Size (MB)	RAM Usage (MB)
XGBoost	24.3	156
Random Forest	156.7	892
CNN-LSTM	42.1	278
CAF-Net	38.6	245

4.4.6 Scalability across Dataset Sizes

We evaluated CAF-Net on progressively larger subsets of CIC-IDS2018 (10%, 25%, 50%, 75%, 100%). Fig. 21 shows that training time scales linearly ($R^2 = 0.994$), while inference latency remains constant.

4.4.7 Accuracy–Latency Trade-Off

Fig. 22 plots F1-score vs. inference latency for all models on CIC-IDS2018. CAF-Net is Pareto-optimal, providing the best accuracy among sub-millisecond latency models.

4.4.8 Practical Deployment Implications

The computational scores given in Sections 4.4.1–4.4.6 are comprehensive for assessing the deployability of CAF-Net in real cyber security applications.

Real-Time Detection Capability: CAF-Net can infer in real-time, as it can process the network traffic of multiple samples per second on a typical cloud infrastructure (AWS `c5.4xlarge`) with an average latency of 0.85 ms per sample. As an example of context, a typical enterprise network that generates 1 Gbps of traffic can generate around 10000–50000 packets per second, thus requiring to deploy on moderate hardware that needs to be parallelized, which is common in production IDS deployments.

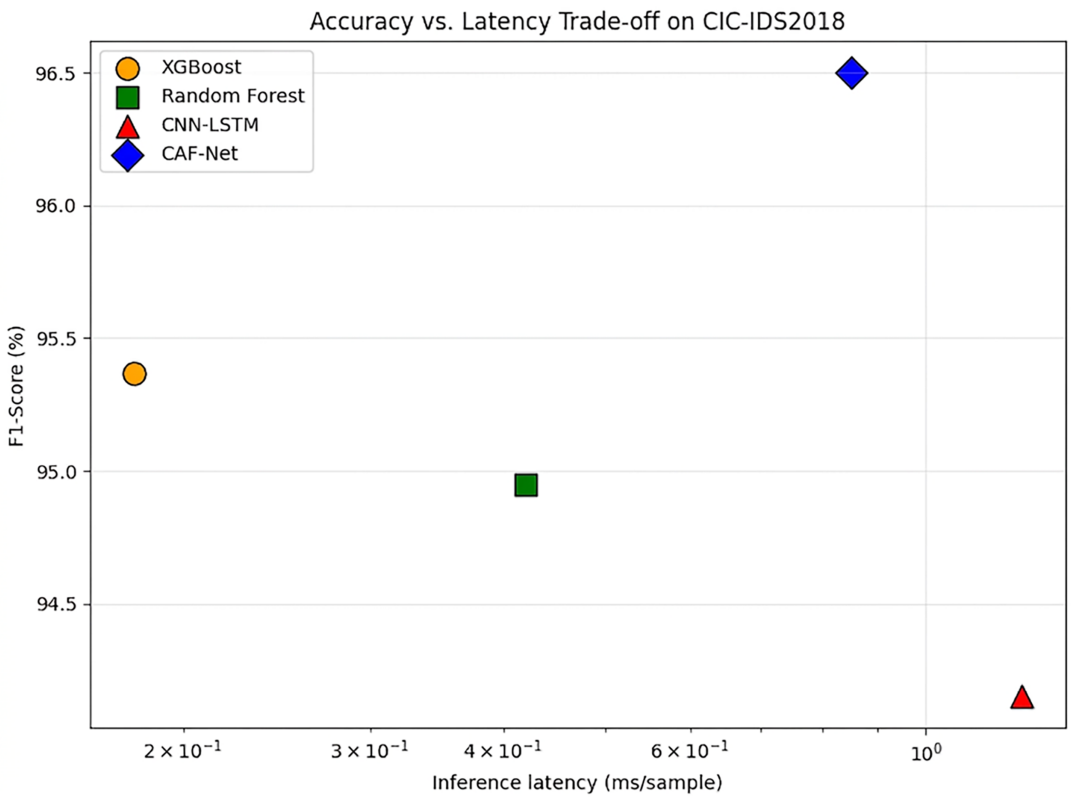


Figure 22: Accuracy–latency pareto frontier. CAF-Net (green star) dominates the trade-off space.

Deployment Scalability: The linear scaling relationship (R^2 = 0.994) of the training time with the number of samples in the data set in Fig. 21 suggests that the training time can be reduced linearly with the size of the data set, which means that as more attacking patterns enter into the data set, CAF-Net can be incrementally retrained with no exponential growth in the training time. Additionally, the inference latency does not vary with different datasets as shown in Fig. 21 depicting it as a function of the training dataset size, indicating that real-time performance is not dependent on the amount of historical training data.

Resource-Constrained Environments: CAF-Net can be deployed on resource-constrained edge devices like network appliances or IoT gateways, as it has a low memory footprint (38.6 MB on disk, 245 MB in memory per deployment). CAF-Net's low memory footprint (38.6 MB disk, 245 MB memory per deployment) makes it suitable for deployment to resource-constrained edge devices such as network appliances or IoT gateways. It is a great benefit compared to deep learning based IDS solutions (e.g., CNN–LSTM solution, 42.1 MB disk space and 278 MB RAM in Table 24) and yet offers greater detection accuracy.
Accuracy–Latency Trade-off: As shown in Fig. 22, the highest accuracy (96.50% F1-score for CIC-IDS2018) in terms of inference latency is achieved by CAF-Net among models with sub-millisecond inference latency. The Pareto-optimal positioning is evidence that CAF-Net achieves a good tradeoff between the detection accuracy and computational efficiency and thus is a viable option for organizations that demand not only security but also low latency times.

Accuracy–Cost Trade-Off Analysis: CAF-Net achieves 96.50% F1 vs. XGBoost's 95.37% (+1.13% improvement) but is 4.7× slower (0.85 vs. 0.18 ms) and uses 57% more RAM (245 MB vs. 156 MB). This is a tradeoff that can be accepted in (i) high security areas where missed detections may be costly, (ii) moderate

throughput (<1200 samples/second), or (iii) areas that have enough memory (>256 MB RAM). XGBoost is a better option if ultra-low latency (<0.5 ms) is demanded, there is limited memory (<200 MB), or if the traffic patterns are simple. For security applications, CAF-Net provides a flexible and high accuracy feature, while the lightness of XGBoost provides a great choice for latency constrained environments.

Enterprise Deployment Recommendations: From the empirical results, CAF-Net is suggested for Enterprise Deployment:

- *High-throughput data centers:* ≈1176 samples/second throughput with latency <1 ms.
- *Resource-constrained edge devices:* 38.6 MB disk footprint and 245 MB RAM usage.
- *Continuous learning environments:* linear training-time scaling for incremental updates (Fig. 21).

These deployment characteristics, combined with the detection accuracy demonstrated in Sections 4.1 and 4.2, establish CAF-Net as a practical and scalable solution for modern network intrusion detection.

4.5 Comparison with Existing Studies

The aim of this unit is to compare existing studies with the information presented. We also validate the effectiveness of our model by comparing the detection accuracy with the results of the related work presented by other studies with various classification techniques. Al-Safi et al. [13] suggested a hybrid approach of Information Gain based feature selection and an Artificial Bee Colony-Cuckoo Search (ABC-CS) optimization technique for optimizing the Support Vector Machine (SVM) parameters. In this proposed algorithm, Information gain based feature selection was incorporated to select the most informative features. Further, the ABC-CS algorithm fine-tunes them selected features to optimize the parameters of the SVM to enhance the classification accuracy. It obtained a maximum accuracy of 94.21, with a recall of 98.25%, precision of 94.99%, and F1 score at 96.59%. As can be seen in Table 25, that proposed methodology proves superior than the one developed in [13] both in terms of results and the extent and sophistication of the technique used. Instead of feature selection with an optimization algorithm for ABC-CS in the process of parameter tuning of the Support Vector Machine, this work uses a four-stage ensemble learning framework (CAF-Net) to utilize the potential of multiple models toward maximal accuracy and robustness. This approach led to excellent performance measures, considering 99.47 accuracy, 99.47 precision, 99.48 recall, and 99.47 F1 score, which showed excellent precision and recall along with both having a great balance. The results, in comparison with leading research on this topic, indicate a considerable improvement of our method concerning the detection of network intrusions. Xu et al. [20] presented an auto-encoder architecture for anomaly detection in network intrusion systems. Their results show that this model performs better than any other related method, and it is especially good for detection of network intrusion with the NSL-KDD dataset. For optimal performance in their proposed model, an auto-encoder architecture consisting of five layers with optimized neuron counts in both the hidden and latent space layers is proposed. In their paper, Mean Absolute Error (MAE) is used as the reconstruction loss function, which gives the best accuracy achieved in the auto-encoder model. Data preprocessing was recognized as a key step whose performance impacts significantly on the model because it has positively improved the quality of the input data. Thus, all the obtained evaluation and performance metrics of [20] were 90.61 accuracy, 86.83 precision, 98.43 recall, and 92.26 F1-score. Reference [20] focuses on the single auto-encoder structure optimized for feature reconstruction, while our model includes an ensemble of several classifiers, which lets the system capture broader patterns and their interactions within the data. The net effect of this multi-model model is improving the general performance of our predictive accuracy and robustness of the system over a single auto-encoder model presented in Table 25. Gurung et al. [22] investigated deep learning techniques for real-time anomaly detection using the NSL-KDD dataset. Their methodology employed max-min normalization, followed by 1-N encoding of the protocol, service, and flag features, resulting in a 115-dimensional representation. An

autoencoder was then combined with Logistic Regression for binary intrusion classification. The model achieved 87.20% accuracy, 84.60% precision, 92.80% recall, and an F1-score of 88.50%. In comparison, the proposed CAF-Net achieved 99.47% accuracy, 99.47% precision, 99.48% recall, and a 99.47% F1-score, demonstrating substantial improvements over the approach of Gurung et al. [22].

Table 25: Comparison of proposed model's performance with related existing studies.

Method	Accuracy	Precision	Recall	F1 Score
[20]	90.61	86.83	98.43	92.26
[22]	87.20	84.60	92.80	88.50
[13]	94.21	94.99	98.25	96.59
[24]	90.73	86.38	93.17	89.65
Proposed Model	99.47	99.47	99.48	99.47

Gurung et al. [22] proposed approach comprise of accuracy of 87.2, precision of 84.6, recall of 92.8, specificity of 80.7 along with a negative predictive value of 90.7. F1 score of the model calculated based on precision and recall are about 88.5. The methodology employed in this methodology was most appropriate to methods applied to meet a deep learning objective; it was to perform better on detecting network intrusion, and it also exhibited strength and ability to accommodate evolving threats. In comparison to using a sparse auto-encoder with logistic regression in the case of [22], our model has tremendous improvements in both the encoding and general performance. Instead, the auto-encoder model relies on 1-N encoding, as in the case of symbolic features, which would give 115 features; this set of features would then enable high dimensionality to be introduced automatically, with even the potential of sparsity. We use our model with label encoding which transformed categorical variables into numeric representation with a smaller representation thereby causing the dimension to remain small and potentially alleviating sparsity. We focus on 15 relevant features chosen with high relevance and information gain to improve the model's efficiency and interpretability. These enhancements result in excellent performance metrics for our model, which incidentally provides 99.47% accuracy, 99.47 precision, 99.48 recall, and 99.47 F1 score compared with an accuracy of 87.2, precision of 84.6, and recall of 92 of the auto-encoder model. Simplified encoding approach and focused feature extraction in the proposed model are some of the key improvements it makes toward the effectiv eness of its potential in network intrusion detection. Fu et al. [24] propose DLNID, a new intrusion detection system using a combined approach to solve issues of class imbalance as well as feature enhancement. The proposed approach integrates ADASYN with auto-encoders, CBAM, and BiLSTM. The use of synthetic samples by ADASYN for the minority class compels the classifier to focus more on more difficult samples. Auto-encoders with dropout have been applied for dimensionality reduction, focusing on saving the most important features and avoiding overfitting. The enhancement that CBAM has upon feature extraction is through the adjustment of channel weights, whereas Bi-LSTM captures the temporal dependencies in the network traffic. The architecture incorporates all these aspects into a deep learning framework to which NSL-KDD dataset was used for testing purposes. The above results in Table 25, clearly indicate its efficiency as accuracy at 90.73%, precision at 86.38%, recall at 93.17%, and F1 score at 89.65% for binary classification. On the other hand, multiclass classifier has been evaluated with accuracy of 85.50%, precision at 86.30%, recall at 84.70%, and the overall F1 score of 85.49%–remarkable on network intrusion detection. Thus in Table 25 it becomes apparent that our method is superior to [24] on the early feature selection and ensemble learning with which class imbalance in the previous case was handled using synthetic data, but our method uses StandardScaler for feature normalization, which enhances model performance along with class imbalance handling.

These findings demonstrate that ensemble approaches outperform more antiquated techniques from earlier studies in terms of network intrusion detection as demonstrated in Fig. 23 It confirms that ensemble learning is very effective for modern intrusion detection systems.

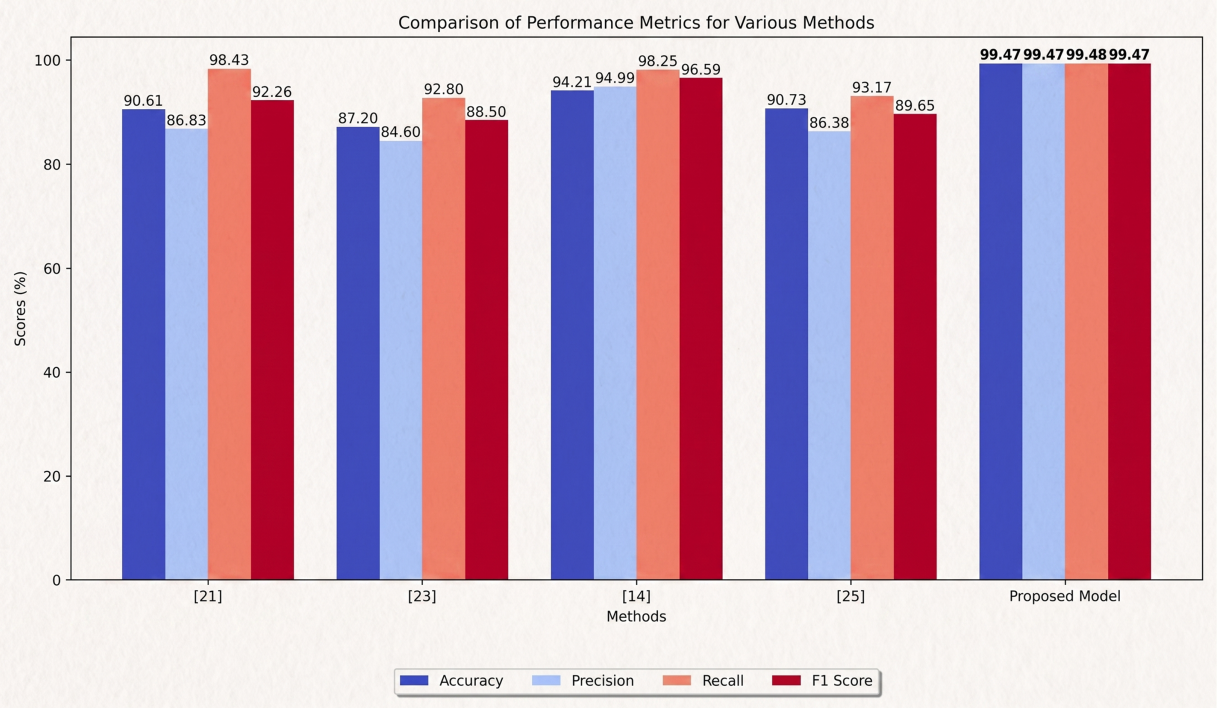


Figure 23: Comparative performance evaluation of the proposed method and related studies.

5 Conclusion and Future Works

In this paper, CAF-Net was introduced, a Context-Aware Fusion Network, that is aimed at solving some of the most critical issues facing contemporary intrusion detection systems, such as dimensionality, imbalance in the number of classes, and reduced flexibility of traditional networks. CAF-Net follows a four-stage ensemble learning framework (preceded by data preprocessing and mutual information-based feature selection), to provide the ability to attain stability and at the same time be expressive in learning. Stage 1 applies bagging with XGBoost to minimize variance and improve robustness. Stage 2 performs heterogeneous stacking by incorporating K-Nearest Neighbors and Logistic Regression to capture complementary decision patterns. Stage 3 introduces an attention-based meta-learning component that dynamically assigns instance-specific importance to base learners to make adaptive and context-sensitive decisions. Stage 4 applies confidence-aware soft voting, increasing the reliability and interpretability of the resulting predictions by fusing probabilistic outputs using both attention relevance and model trustworthiness. Large-scale experimental analyses of benchmark data, such as NSL-KDD and CIC-IDS2018, confirm that CAF-Net performs well when compared to baseline and ensemble. The model achieves good detection accuracy and a good balance between precision and recall, even in severe conditions with the presence of imbalanced classes. Ablation experiments also confirm the validity of the proposal, finding that every component feature selection, attention mechanism and soft voting makes a significant contribution to overall performance, validating the effectiveness of the proposed design. Overall, CAF-Net is an effective, scalable, and interpretable multimodal network intrusion detection platform that can capture complex and diverse patterns of network traffic. It can be practically deployed with extensive computational metrics such as

inference latency of 0.85 ms per sample (1176 samples/second throughput), linear training time scalability (R^2 = 0.994), and a small memory footprint of 38.6 MB disk and 245 MB RAM memory. The attributes of CAF-Net along with its outstanding detection capabilities on various benchmark datasets make it a potential contender for deployment in real-world cybersecurity applications such as enterprise network environments with high throughput and edge devices with limited computational resources. But there are some things to take into account. This is because NSL-KDD has established patterns of performance which is not to say that this will translate to similar performance in all operational environments. Unlike a set of benchmark datasets, real world intrusion detection is defined by changing attacks, different types of network architecture, and varying business constraints. The score on the more challenging CIC-IDS2018 and UNSW-NB15 datasets is evidence of generalization, however, for deployment in production environments validation on domain-specific data would be needed, integration with the existing security infrastructure would be required, and adaptation to operational requirements, such as specific latency and throughput rules, would be necessary. Future areas of research are expanding the framework to include deep representation learning for detecting attacks when they occur for the first time, optimizing inference times for high traffic volumes, adding explanations and other AI techniques to enable security analysts to the system to provide decisions, and testing the framework in operational network setups with various attack patterns and traffic volumes. Such extensions will further reinforce the practical value of CAF-Net, and will help to overcome the drawbacks of the benchmark driven evaluation approach.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Venkata Mohit Tamanampudi, Zaid Bin Faheem, Jehad Ali; methodology development: Venkata Mohit Tamanampudi, Zaid Bin Faheem, Jehad Ali; data collection and preprocessing: Venkata Mohit Tamanampudi, Zaid Bin Faheem; formal analysis and investigation: Venkata Mohit Tamanampudi, Zaid Bin Faheem, Jehad Ali; software implementation and validation: Venkata Mohit Tamanampudi, Zaid Bin Faheem; writing—original draft preparation: Venkata Mohit Tamanampudi, Zaid Bin Faheem; writing—review and editing: Venkata Mohit Tamanampudi, Zaid Bin Faheem, Jehad Ali; supervision and project administration: Jehad Ali. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in public repositories. The NSL-KDD dataset is available from Kaggle at https://www.kaggle.com/datasets/hassan06/nslkd. The UNSW-NB15 and CIC-IDS2018 datasets are publicly available through their respective official sources. The code and implementation details for the proposed CAF-Net framework are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: As an Editorial Board Member of this journal, [Jehad Ali] had no involvement in the peer review of this article and had no access to information regarding its peer review. Full responsibility for the editorial process for this article was delegated to another journal editor. The authors declare no other conflicts of interest.

References

1. Bhavsar YB, Waghmare KC. Intrusion detection system using data mining technique: support vector machine. Int J Emerg Technol Adv Eng. 2013;3(3):581–6. doi:10.1016/j.eij.2013.10.003.
2. Rastogi S, Shrotriya A, Singh MK, Potukuchi RV. An analysis of intrusion detection classification using supervised machine learning algorithms on NSL-KDD dataset. J Comput Res Innov. 2022;7(1):124–37. doi:10.24191/jcrinn.v7i1.274.

3. Repalle SA, Kolluru VR. Intrusion detection system using AI and machine learning algorithm. Int Res J Eng Technol. 2017;4(12):1709–15. doi:10.55248/gengpi.07.0426.20825.
4. Ting KM. Precision and recall. In: Sammut C, Webb GI, editors. Encyclopedia of machine learning. Boston, MA, USA: Springer US; 2011. p. 781. doi:10.1007/978-0-387-30164-8_652.
5. Govindarajan V, Ahmed F, Faheem ZB, Bilal M, Ayadi M, Ali J. Aegis-5: a hybrid ensemble framework for intrusion detection in industry 5.0 driven smart manufacturing environment. ACM Trans Auton Adapt Syst. 2026. doi:10.1145/378722.
6. Revathi S, Malathi A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. Int J Eng Res Technol. 2013;2(12):1848–53. doi:10.1007/s44163-026-01930-9.
7. Liu J, Kantarci B, Adams C. Machine learning-driven intrusion detection for Contiki-NG-based IoT networks exposed to NSL-KDD dataset. In: Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning; 2020 Jul 13; Virtual. p. 25–30. doi:10.1145/3395352.3402621.
8. Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 2017;5:21954–61. doi:10.1109/access.2017.2762418.
9. Chou D, Jiang M. A survey on data-driven network intrusion detection. ACM Comput Surv. 2021;54(9):1–36. doi:10.1145/3472753.
10. Buczak AL, Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor. 2015;18(2):1153–76. doi:10.1109/comst.2015.2494502.
11. Ma T, Wang F, Cheng J, Yu Y, Chen X. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors. 2016;16(10):1701. doi:10.3390/s16101701.
12. Raj S, Jain M, Chouksey P. A network intrusion detection system based on categorical boosting technique using NSL-KDD. Indian J Cryptogr Netw Secur. 2021;1(2):1–4. doi:10.35940/ijcns.b1411.111221.
13. Al-Safi AHS, Hani ZIR, Zahra MMA. Using a hybrid algorithm and feature selection for network anomaly intrusion detection. J Mech Eng Res Dev. 2021;44(4):253–62.
14. Zakariah M, AlQahtani SA, Alawwad AM, Alotaibi AA. Intrusion detection system with customized machine learning techniques for NSL-KDD dataset. Comput Mater Contin. 2023;77(3):4025–54. doi:10.32604/cmc.2023.043752.
15. Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F. Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans Emerg Telecommun Technol. 2021;32(1):e4150.
16. Ghurab M, Gaphari G, Alshami F, Alshamy R, Othman S. A detailed analysis of benchmark datasets for network intrusion detection system. Asian J Res Comput Sci. 2021;7(4):14–33. doi:10.9734/ajrcos/2021/v7i430185.
17. Patil DR, Pattewar TM. A comparative performance evaluation of machine learning-based NIDS on benchmark datasets. Int J Res Advent Technol. 2014;2(2):101–6.
18. Al-mamory SO, Jassim FS. Evaluation of different data mining algorithms with KDD CUP 99 data set. J Babylon Univ Pure Appl Sci. 2013;21(8):2663–81.
19. Hu J, Liu C, Cui Y. An improved CNN approach for network intrusion detection system. Int J Netw Secur. 2021;23(4):569–75. doi:10.47277/ijcncs/2(2)4.
20. Xu W, Jang-Jaccard J, Singh A, Wei Y, Sabrina F. Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset. IEEE Access. 2021;9:140136–46. doi:10.1109/access.2021.3116612.
21. Almutairi YS, Alhazmi B, Munshi AA. Network intrusion detection using machine learning techniques. Adv Sci Technol Res J. 2022;16(3):193–206. doi:10.12913/22998624/149934.
22. Gurung S, Ghose MK, Subedi A. Deep learning approach on network intrusion detection system using NSL-KDD dataset. Int J Comput Netw Inf Secur. 2019;11(3):8–14. doi:10.5815/ijcnis.2019.03.02.
23. Alzahrani AO, Alenazi MJF. Designing a network intrusion detection system based on machine learning for software defined networks. Future Internet. 2021;13(5):111. doi:10.3390/fi13050111.
24. Fu Y, Du Y, Cao Z, Li Q, Xiang W. A deep learning model for network intrusion detection with imbalanced data. Electronics. 2022;11(6):898. doi:10.3390/electronics11060898.
25. Jo W, Kim S, Lee C, Shon T. Packet preprocessing in CNN-based network intrusion detection system. Electronics. 2020;9(7):1151. doi:10.3390/electronics9071151.

26. Magdy ME, Matter AM, Hussin S, Hassan D, Elsaid S. A comparative study of intrusion detection systems applied to NSL-KDD dataset. Egypt Int J Eng Sci Technol. 2023;43(2):88–98. doi:10.21608/eijest.2022.137441.1156.
27. Niyaz Q, Sun W, Javaid AY, Alam M. A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (BICT); 2015 Dec 3–5; New York, NY, USA. p. 1–6. doi:10.4108/eai.3-12-2015.2262516.
28. Patel S, Sondhi J. A review of intrusion detection technique using various technique of machine learning and feature optimization technique. Int J Comput Appl. 2014;93(14):43–7. doi:10.5120/16286-6185.
29. Staudemeyer RC, Omlin CW. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In: Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT'13); 2013 Oct 7–9; East London, South Africa. p. 218–24.
30. Bansal A, Mahapatra S. A comparative analysis of machine learning techniques for botnet detection. In: Proceedings of the 10th International Conference on Security of Information and Networks (SIN'17); 2017 Oct 13–15; Jaipur, India. p. 91–8.
31. Mohammed B, Gbashi EK. Intrusion detection system for NSL-KDD dataset based on deep learning and recursive feature elimination. Eng Technol J. 2021;39(7):1069–79. doi:10.30684/etj.v39i7.1695.
32. Alavizadeh H, Alavizadeh H, Jang-Jaccard J. Deep Q-learning based reinforcement learning approach for network intrusion detection. Computers. 2022;11(3):41. doi:10.3390/computers11030041.
33. Sindhu SS, Geetha S, Kannan A. Decision tree based light weight intrusion detection using a wrapper approach. Expert Syst Appl. 2012;39(1):129–41. doi:10.1016/j.eswa.2011.06.013.
34. Su T, Sun H, Zhu J, Wang S, Li Y. BAT: deep learning methods on network intrusion detection using NSL-KDD dataset. IEEE Access. 2020;8:29575–85. doi:10.1109/access.2020.2972627.
35. Patel A, Taghavi M, Bakhtiyari K, Júnior JC. An intrusion detection and prevention system in cloud computing: a systematic review. J Netw Comput Appl. 2013;36(1):25–41. doi:10.1016/j.jnca.2012.08.007.
36. Tang Y, Gu L, Wang L. Deep stacking network for intrusion detection. Sensors. 2022;22(1):25. doi:10.3390/s22010025.
37. Zhao N, Lun D. Composite anti-disturbance control for networked systems with disturbances and actuator attacks via event-triggered output feedback. IEEE Trans Cybern. 2025;56(1):393–403. doi:10.1109/tcyb.2025.3609819.
38. Sun H, Peng C, Zhang W, Yang T, Wang Z. Security-based resilient event-triggered control of networked control systems under denial of service attacks. J Frankl Inst. 2019;356(17):10277–95. doi:10.1016/j.jfranklin.2018.04.001.
39. Zhao N, Shi P, Xing W, Lim CP. Event-triggered control for networked systems under denial of service attacks and applications. IEEE Trans Circuits Syst I Regul Pap. 2022;69(2):811–20. doi:10.1109/tcsi.2021.3116278.
40. Suthaharan S. Big data classification: problems and challenges in network intrusion prediction with machine learning. New York, NY, USA: ACM; 2014.
41. Dhanabal L, Shantharajah SP. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. Int J Adv Res Comput Commun Eng. 2015;4(6):446–52.
42. ZAIB MH, Kaggle. NSL-KDD dataset. 2024 [cited 2024 Dec 15]. Available from: https://www.kaggle.com/datasets/hassan06/nslkdd.
43. Iftikhar N, Rehman MU, Shah MA, Alenazi MJF, Ali J. Intrusion detection in NSL-KDD dataset using hybrid self-organizing map model. Comput Model Eng Sci. 2025;143(1):639–71. doi:10.32604/cmes.2025.062788.
44. Sherin VIJ, Radhika N. Stacked ensemble-IDS using NSL-KDD dataset. J Pharm Negat Results. 2022;13:351–6. doi:10.47750/pnr.2022.13.S03.057.
45. Ali J, Song HH, Roh BH. An SDN-based framework for E2E QoS guarantee in Internet of Things devices. IEEE Internet Things J. 2024;12(1):605–22. doi:10.1109/jiot.2024.3465609.
46. Tumkur SD, Eswarakrishnan V, Wairagade A, Aslam MA, Bilal M, Cheema AM. Optimizing malware detection in virtual cloud environments using hybrid machine learning approach. Arab J Sci Eng. 2025;50(23):19463–94. doi:10.1007/s13369-025-10205-x.
47. Dhara S. Data pre-processing and feature engineering for optimization. In: Artificial intelligence techniques in mathematical modeling and optimization. Boca Raton, FL, USA: CRC Press; 2026. p. 32–56.

48. Kanstren T. A look at precision, Recall, and F1-Score. [cited 2026 Jan 1]. Available from: https://medium.com/data-science/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec.
49. Sokolova M, Japkowicz N, Szpakowicz S. Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In: Australasian Joint Conference on Artificial Intelligence. Berlin/Heidelberg, Germany: Springer; 2006
50. Hernández-Del-Toro T, Martínez-Santiago F, Montejo-Ráez A. Assessing classifier's performance. In: Biosignal processing and classification using computational learning and intelligence. Amsterdam, The Netherlands: Elsevier; 2022. p. 131–49.
51. Rajagopal S, Kundapur PP, Hareesha KS. A stacking ensemble for network intrusion detection using heterogeneous datasets. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2020.