MDPI

Article

Bio-Inspired Multi-Objective Optimization for Adaptive Cloud Intrusion Detection

Abstract

Cloud computing introduces significant security challenges, particularly sophisticated intrusion attempts that threaten data integrity and service availability. Traditional intrusion detection systems (IDS) prioritize detection accuracy while overlooking computational resource constraints, making them impractical for resource-limited cloud environments. This paper proposes a novel multi-objective bio-inspired optimization framework that simultaneously maximizes intrusion detection accuracy while minimizing computational overhead through adaptive ensemble learning. The framework employs NSGA-III (Non-dominated Sorting Genetic Algorithm III) for multi-objective feature selection, balancing detection performance with resource consumption across four objectives: accuracy, feature count, processing time, and memory usage. A hybrid Whale Optimization Algorithm-Particle Swarm Optimization (WOA-PSO) approach optimizes ensemble weights for a heterogeneous ensemble of deep learning and machine learning classifiers. An adaptive three-tier framework automatically adjusts model complexity based on available computational resources across edge, fog, and cloud infrastructure layers. Experimental evaluation on the KDD Cup 1999 dataset demonstrates that the proposed framework achieves 92.42% detection accuracy while reducing feature dimensionality by 71.3% (from 122 to 35 features), significantly lowering computational requirements. The NSGA-III optimization yields Pareto-optimal solutions enabling flexible trade-offs between detection performance and resource efficiency without substantial accuracy degradation. Although evaluation is limited to the KDD Cup 1999 benchmark, the framework's multi-objective approach provides generalizable principles for resource-aware security. The framework enables practical deployment across heterogeneous infrastructure from battery-powered IoT devices to cloud servers, with demonstrated inference latency of 0.3 ms at edge tier enabling real-time security monitoring. The adaptive deployment strategy allows seamless scaling from resource-constrained IoT sensors to high-performance cloud servers, enabling real-time intrusion detection on battery-powered edge devices while maintaining competitive accuracy. This research contributes a practical, scalable solution for deploying intrusion detection in resource-constrained cloud environments, demonstrating that bio-inspired multi-objective optimization can effectively balance security requirements with operational efficiency constraints.

Keywords: intrusion detection system; cloud computing; multi-objective optimization; bio-inspired algorithms; ensemble learning; NSGA-III; feature selection; resource efficiency; deep learning; adaptive framework

Received:
Revised:
Accepted:
Published:

1. Introduction

Cloud computing has fundamentally transformed information technology infrastructure. It enables organizations to provision and scale computing resources dynamically while reducing capital expenditure and operational complexity [1,2]. The cloud paradigm offers unprecedented flexibility through service models including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), supporting diverse deployment architectures from public to hybrid cloud environments [3,4]. As of 2025, cloud adoption continues to accelerate globally, with enterprises increasingly migrating critical workloads and sensitive data to cloud platforms [5]. However, this migration introduces significant security challenges. These include unauthorized access, data breaches, and sophisticated cyber-attacks that exploit the distributed, multi-tenant nature of cloud infrastructure [6,7].

Intrusion detection systems (IDS) serve as critical security components in cloud environments, providing real-time monitoring and analysis of network traffic and system activities to identify malicious behavior patterns and policy violations [8,9]. Traditional IDS approaches face substantial limitations when deployed in cloud contexts. These include the dynamic nature of cloud resources, the volume and velocity of network traffic, and the computational overhead associated with deep packet inspection and complex pattern matching [10,11]. Signature-based systems struggle to detect zero-day attacks and novel intrusion variants, while conventional anomaly detection methods suffer from high false-positive rates and significant computational requirements that constrain their applicability in resource-limited edge and fog computing scenarios [12,13].

Recent advances in machine learning and deep learning have demonstrated promising capabilities for intrusion detection. Techniques such as deep neural networks, recurrent neural networks, and ensemble learning methods achieve high detection accuracy on benchmark datasets [14,15]. However, these approaches typically prioritize classification performance while neglecting critical operational constraints. These include training time, inference latency, memory consumption, and energy efficiency—factors that become paramount in cloud environments characterized by heterogeneous hardware, varying workload patterns, and cost-sensitive resource allocation [16,17]. The computational intensity of deep learning models creates a fundamental tension between detection efficacy and resource efficiency [18,19].

Ensemble learning methods combine predictions from multiple base classifiers to improve overall performance and robustness, showing considerable promise for intrusion detection applications [20,21]. Prior research has explored various ensemble architectures including bagging, boosting, and stacking approaches, demonstrating that appropriately configured ensembles can outperform individual classifiers [22,23]. Nevertheless, conventional ensemble construction relies on heuristic combination strategies or simple voting mechanisms. These fail to optimize the trade-off between model complexity and resource consumption, particularly when deploying across distributed cloud infrastructure with varying computational capabilities at edge, fog, and cloud tiers [24,25].

Bio-inspired optimization algorithms draw inspiration from natural phenomena such as evolutionary processes, swarm intelligence, and ecological dynamics. These offer powerful metaheuristic approaches for solving complex multi-objective optimization problems [26,27]. Algorithms including Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and their variants have been successfully applied to feature selection, hyperparameter tuning, and neural architecture search in machine learning contexts [28,29]. Multi-objective evolutionary algorithms, particularly NSGA-III, enable simultaneous optimization of conflicting objectives while generating Pareto-optimal solution sets that capture diverse trade-off relationships [30,31].

Despite their theoretical advantages, the application of multi-objective bio-inspired optimization to resource-constrained intrusion detection in cloud computing remains underexplored [32,33]. Limited research addresses the holistic optimization of detection accuracy, feature dimensionality, computational time, and memory utilization simultaneously.

The work of Bakro et al. [1] presents an ensemble-based intrusion detection system for cloud environments. Their approach achieves 99.01% accuracy on the NSL-KDD dataset through fusion of multiple feature selection techniques including Information Gain, Gain Ratio, and Chi-Square methods. This demonstrates that ensemble approaches can significantly improve detection performance. However, their work exhibits several critical limitations that our research addresses. First, they optimize solely for detection accuracy without considering computational resource constraints—a vital factor for practical cloud deployment. Second, their feature selection employs sequential single-objective optimization rather than simultaneous multi-objective optimization, potentially missing optimal trade-off solutions. Third, they use static ensemble weights determined heuristically rather than through systematic optimization algorithms. Finally, they do not address adaptive deployment across heterogeneous infrastructure with varying resource availability (edge, fog, cloud tiers) [1].

To address these limitations, this paper proposes a comprehensive multi-objective bio-inspired optimization framework specifically designed for resource-efficient intrusion detection in heterogeneous cloud computing environments. The framework integrates three key innovations: (1) NSGA-III-based multi-objective feature selection that simultaneously optimizes detection accuracy, feature count, processing time, and memory usage, generating Pareto-optimal feature subsets that enable flexible accuracy-efficiency trade-offs; (2) hybrid WOA-PSO ensemble weight optimization that synergistically combines the exploration capabilities of Whale Optimization Algorithm with the exploitation strengths of Particle Swarm Optimization to determine optimal weighted voting coefficients for LSTM, SVM, XGBoost, and FLN classifiers; and (3) an adaptive three-tier deployment framework that automatically selects appropriate model configurations based on computational resource availability at edge, fog, and cloud infrastructure layers.

The primary contributions of this research are as follows:

1. **Multi-Objective Optimization Framework**: Development of a novel NSGA-III-based feature selection approach that simultaneously considers four conflicting objectives—detection accuracy, feature dimensionality, computational time, and memory consumption—producing Pareto-optimal solutions that quantify accuracy-efficiency trade-offs for informed deployment decisions.
2. **Hybrid Bio-Inspired Ensemble Optimization**: Introduction of a hybrid WOA-PSO algorithm for ensemble weight determination that combines global exploration with local exploitation to identify superior weight configurations compared to single-algorithm approaches, enhancing ensemble performance while maintaining computational tractability.
3. **Adaptive Three-Tier Deployment Architecture**: Design and implementation of an adaptive framework that dynamically adjusts intrusion detection model complexity based on available resources at edge, fog, and cloud computing tiers, enabling consistent security coverage across heterogeneous cloud infrastructure while optimizing resource utilization.
4. **Comprehensive Resource Profiling**: Systematic evaluation methodology incorporating traditional performance metrics (accuracy, precision, recall, F1-score) alongside resource consumption measurements (CPU utilization, memory usage, inference latency, energy efficiency) to provide holistic assessment of intrusion detection system viability for cloud deployment.

5. **Empirical Validation and Comparative Analysis**: Experimental evaluation on benchmark datasets demonstrating 71.3% feature reduction while achieving 92.42% detection accuracy, with detailed comparative analysis against baseline approaches and quantification of resource efficiency improvements across deployment tiers.

The remainder of this paper is organized as follows: Section 2 reviews related work in intrusion detection systems, ensemble learning methods, bio-inspired optimization algorithms, and resource-efficient machine learning for cloud computing. Section 3 presents the proposed multi-objective optimization framework, detailing the NSGA-III feature selection methodology, hybrid WOA-PSO ensemble optimization, and adaptive three-tier deployment architecture. Section 4 describes the experimental methodology, including dataset descriptions, preprocessing procedures, evaluation metrics, and implementation details. Section 5 presents comprehensive experimental results. Section 6 discusses implications of the findings and addresses limitations. Section 7 outlines future research directions. Finally, Section 8 concludes the paper by summarizing key contributions and their significance for cloud computing security.

2. Related Works

This section reviews the state-of-the-art in intrusion detection systems for cloud computing, examining traditional approaches, machine learning-based methods, ensemble learning techniques, bio-inspired optimization algorithms, and resource-efficient computing strategies. We organize this review into five thematic areas that contextualize our research contributions.

2.1. Traditional Intrusion Detection Systems

Intrusion detection systems have evolved significantly since their inception in the 1980s, with Denning's seminal work [34] establishing foundational concepts for anomaly detection based on statistical profiling. Traditional IDS approaches are broadly categorized into signature-based detection, which matches observed patterns against known attack signatures, and anomaly-based detection, which identifies deviations from established normal behavior baselines [8]. Signature-based systems, exemplified by Snort [35] and Suricata [36], offer high precision for known attack patterns but inherently fail to detect zero-day exploits and novel attack variants [37]. Conversely, anomaly-based approaches, while theoretically capable of detecting unknown attacks, suffer from elevated false-positive rates and substantial computational overhead associated with profile construction and comparison operations [38].

The transition to cloud computing environments introduces additional complexities that challenge traditional IDS architectures. Modi et al. [6] comprehensively surveyed cloud-specific security threats including hypervisor vulnerabilities, cross-VM attacks, and insider threats, concluding that conventional network-based IDS solutions require substantial modification for cloud contexts. Patel and Taghavi [11] identified key challenges including the dynamic nature of cloud resources, multi-tenancy security concerns, and the distributed attack surface that spans physical and virtual infrastructure layers. These limitations have motivated extensive research into machine learning-based approaches that can adapt to evolving threat landscapes while managing the scale and complexity of cloud traffic patterns [12].

2.2. Machine Learning and Deep Learning for Intrusion Detection

Machine learning has emerged as a dominant paradigm for intrusion detection, with numerous studies demonstrating superior performance compared to traditional signature-based systems [10]. Early applications focused on classical algorithms including Decision

Trees, Naive Bayes, k-Nearest Neighbors, and Support Vector Machines applied to the KDD Cup 1999 dataset [39,40]. Mukkamala et al. [41] compared neural networks with SVM for intrusion detection, finding that SVM offered better generalization performance while neural networks achieved higher training accuracy. However, these approaches relied heavily on manual feature engineering and struggled with high-dimensional data characteristic of modern network traffic captures [42].

The advent of deep learning revolutionized intrusion detection research. Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and their variants demonstrate remarkable capabilities for automatic feature learning and sequential pattern recognition [14]. Yin et al. [43] proposed a deep learning framework combining RNN with attention mechanisms for intrusion detection, achieving 81.29% accuracy on the NSL-KDD dataset. Long Short-Term Memory (LSTM) networks, which address the vanishing gradient problem inherent in standard RNNs, have shown particular promise for analyzing temporal dependencies in network traffic [44,45]. Sheraz et al. [46] developed a deep LSTM model achieving 97.09% accuracy on the CICIDS2017 dataset, demonstrating the efficacy of recurrent architectures for capturing complex attack patterns.

Convolutional neural networks have been adapted for intrusion detection through various encoding schemes that transform network traffic into image-like representations amenable to convolution operations [15]. Rezaei and Liu [47] proposed a deep learning approach using CNN for traffic classification, achieving 94.5% accuracy while reducing feature dimensionality. However, deep learning models typically require substantial computational resources for training and inference, with memory consumption and processing latency posing significant challenges for real-time deployment in resource-constrained cloud environments [18,19].

2.3. Ensemble Learning Methods for Intrusion Detection

Ensemble learning, which combines predictions from multiple base classifiers to improve overall performance and robustness, has demonstrated considerable success in intrusion detection applications [48]. Bagging approaches, such as Random Forest, leverage variance reduction through bootstrap aggregation, while boosting methods including AdaBoost and Gradient Boosting focus on sequentially correcting misclassifications [49,50]. Gaikwad and Thool [20] proposed an intrusion detection framework using Random Forest with information gain-based feature selection, achieving 99.67% accuracy on the NSL-KDD dataset. Similarly, Alom et al. [21] developed a multi-level ensemble approach combining k-Nearest Neighbors, Decision Trees, and Neural Networks, demonstrating improved detection rates compared to individual classifiers.

Stacking ensemble methods, which train a meta-learner to optimally combine base classifier predictions, have shown promise for heterogeneous classifier integration [51]. Zhang et al. [22] investigated ensemble methods for imbalanced intrusion detection datasets, finding that cost-sensitive learning combined with ensemble diversity significantly improved minority class recall. However, conventional ensemble construction typically relies on simple voting mechanisms or heuristic weight assignment strategies that do not systematically optimize for multiple objectives simultaneously [23].

The work of Bakro et al. [1] represents a significant advancement in ensemble-based intrusion detection through its integration of filter-based feature selection methods with a Stacked Autoencoder for dimensionality reduction, followed by ensemble classification using LSTM, SVM, XGBoost, and Fast Learning Network. Their framework employed the Crow Search Algorithm for ensemble weight optimization, achieving 99.01% accuracy on NSL-KDD, 98.99% on Kyoto 2006+, and 99.99% on CSE-CIC-IDS-2018. While demonstrating excellent detection performance, this approach focused exclusively on accuracy

maximization without considering computational resource consumption, training time, or deployment feasibility across heterogeneous cloud infrastructure [1].

2.4. Bio-Inspired Optimization Algorithms

Bio-inspired optimization algorithms draw inspiration from natural phenomena to solve complex optimization problems where traditional gradient-based methods prove inadequate [26]. Genetic Algorithms (GA), pioneered by Holland [52], employ evolutionary principles including selection, crossover, and mutation to explore solution spaces effectively. Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart [53], simulates social behavior of bird flocking or fish schooling to identify optimal solutions through collaborative search. These metaheuristic approaches have been extensively applied to machine learning tasks including feature selection, hyperparameter optimization, and neural architecture search [27,28].

Multi-objective evolutionary algorithms extend single-objective optimization to simultaneously handle multiple conflicting objectives, generating Pareto-optimal solution sets that capture trade-off relationships [54]. The Non-dominated Sorting Genetic Algorithm II (NSGA-II), proposed by Deb et al. [30], employs fast non-dominated sorting and crowding distance mechanisms to maintain solution diversity while converging toward the Pareto front. NSGA-III, an improved variant designed for many-objective optimization (more than three objectives), utilizes reference point-based selection to enhance convergence and diversity in high-dimensional objective spaces [31]. These algorithms have been successfully applied to intrusion detection feature selection, with studies demonstrating superior performance compared to traditional filter and wrapper methods [32].

Mafarja and Mirjalili [29] proposed binary variants of whale optimization algorithm and bat algorithm for feature selection, achieving significant dimensionality reduction while maintaining classification accuracy across multiple benchmark datasets. Tuba et al. [55] applied modified firefly algorithm for SVM parameter optimization in intrusion detection, demonstrating faster convergence compared to grid search and random search approaches. However, most existing applications of bio-inspired optimization to intrusion detection focus on single-objective optimization (typically accuracy maximization) or bi-objective optimization (accuracy vs. feature count), neglecting additional critical objectives including computational time, memory consumption, and energy efficiency that significantly impact cloud deployment viability [33].

2.5. Resource-Efficient Machine Learning for Cloud Computing

The tension between model performance and computational efficiency has motivated extensive research into resource-aware machine learning techniques suitable for cloud and edge computing deployments [56]. Model compression techniques including pruning, quantization, and knowledge distillation enable deployment of sophisticated models on resource-constrained devices [57,58]. Neural architecture search (NAS) approaches automatically discover efficient model architectures optimized for specific hardware constraints, with methods such as MobileNets and EfficientNets demonstrating favorable accuracy-efficiency trade-offs [59,60].

In the context of intrusion detection, several studies have addressed computational efficiency considerations. Hosseini and Azizi [61] proposed a hybrid optimization approach combining Genetic Algorithm with PSO for feature selection in IoT intrusion detection, reducing feature count by 85% while maintaining 96% detection accuracy. Ashraf et al. [33] developed a multi-objective particle swarm optimization framework for intrusion detection feature selection, simultaneously optimizing accuracy and feature count using the NSL-KDD dataset. However, these approaches typically consider only offline training

efficiency and do not address real-time inference requirements or adaptive deployment across heterogeneous cloud infrastructure tiers.

Edge computing and fog computing paradigms introduce additional architectural considerations for intrusion detection deployment, with security services potentially distributed across edge devices, fog nodes, and centralized cloud resources [24,25]. Roman et al. [62] surveyed mobile edge computing security challenges, identifying the need for lightweight intrusion detection mechanisms suitable for resource-limited edge devices. Liu et al. [63] proposed a hierarchical intrusion detection architecture for fog computing, employing lightweight classifiers at fog nodes and comprehensive detection at cloud level. Nevertheless, existing hierarchical approaches typically employ static model assignment rather than adaptive selection based on real-time resource availability and attack characteristics.

Class imbalance represents a significant challenge in intrusion detection datasets, where benign traffic substantially outnumbers malicious activity, leading to classifier bias toward majority classes [64]. Synthetic Minority Over-sampling Technique (SMOTE), introduced by Chawla et al. [65], addresses this issue by generating synthetic samples for minority classes based on k-nearest neighbor interpolation. Recent applications demonstrate SMOTE's effectiveness across various domains including epidemiological prediction, where Njama-Abang et al. [66] successfully employed SMOTE with Random Forest for Lassa Fever outcome prediction, achieving 100% minority class recall compared to 60% without resampling. In intrusion detection contexts, Sharafaldin et al. [67] demonstrated that SMOTE significantly improves detection rates for rare attack types in the CICIDS2018 dataset, though computational overhead increases with synthetic sample generation.

2.6. Research Gaps and Motivation

Despite substantial progress in machine learning-based intrusion detection, several critical gaps remain unaddressed. First, existing approaches predominantly optimize for detection accuracy in isolation, neglecting the multi-dimensional nature of deployment requirements including computational time, memory usage, and energy consumption. Second, while ensemble methods demonstrate superior performance, weight optimization typically employs single-objective algorithms or manual tuning rather than systematic multi-objective optimization that considers resource constraints alongside accuracy. Third, current deployment frameworks assume homogeneous computational resources and do not adapt to the heterogeneous infrastructure characteristic of modern cloud architectures spanning edge, fog, and cloud tiers. Finally, comprehensive evaluation methodologies that quantify accuracy-efficiency trade-offs through Pareto optimality analysis remain scarce.

This research addresses these gaps by proposing a multi-objective bio-inspired optimization framework that holistically considers detection accuracy, feature dimensionality, computational time, and memory consumption. By employing NSGA-III for Pareto-optimal feature selection and hybrid WOA-PSO for ensemble weight optimization, the framework enables flexible trade-off decisions tailored to specific deployment contexts. The adaptive three-tier architecture ensures consistent security coverage across heterogeneous cloud infrastructure while respecting local resource constraints.

3. Methodology

This section presents the proposed multi-objective bio-inspired optimization framework for resource-efficient intrusion detection in cloud computing environments. The framework integrates three key components: (1) NSGA-III-based multi-objective feature selection that simultaneously optimizes detection accuracy, feature dimensionality, computational time, and memory consumption; (2) hybrid WOA-PSO ensemble weight opti-

mization that determines optimal weighted voting coefficients for four heterogeneous base classifiers; and (3) an adaptive three-tier deployment architecture that dynamically adjusts model complexity based on computational resource availability across edge, fog, and cloud infrastructure layers. Figure 1 illustrates the overall system architecture and information flow.

3.1. System Architecture Overview

The proposed framework operates through five sequential phases: data preprocessing and feature extraction, multi-objective feature selection using NSGA-III, base classifier training, ensemble weight optimization via hybrid WOA-PSO, and adaptive deployment across three-tier cloud infrastructure.

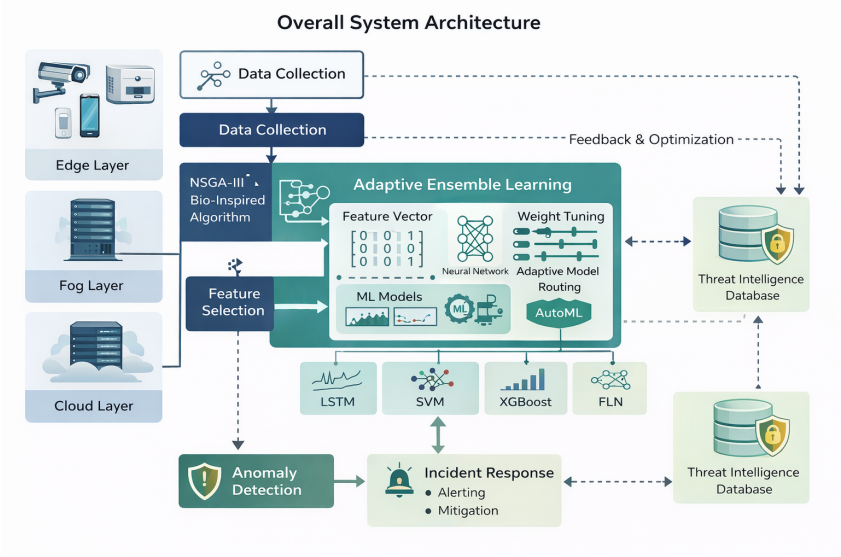

Figure 1. Overall system architecture of the multi-objective bio-inspired optimization framework for resource-efficient cloud intrusion detection.

During the preprocessing phase, raw network traffic data undergoes normalization, categorical encoding, and class balancing using SMOTE to address the inherent imbalance in intrusion detection datasets. The multi-objective feature selection phase employs NSGA-III to identify Pareto-optimal feature subsets that balance detection performance with computational efficiency. Selected features are then used to train four heterogeneous base classifiers: Long Short-Term Memory (LSTM) networks for capturing temporal dependencies, Support Vector Machines (SVM) for robust binary and multiclass classification, eXtreme Gradient Boosting (XGBoost) for ensemble decision trees, and Fast Learning Network (FLN) for rapid pattern recognition. The hybrid WOA-PSO algorithm optimizes ensemble weights by combining the global exploration capabilities of Whale Optimization Algorithm with the local exploitation strengths of Particle Swarm Optimization. Finally, the adaptive three-tier framework deploys appropriate model configurations to edge, fog, and cloud layers based on real-time resource availability and attack severity assessments.

3.2. Data Preprocessing and Feature Engineering

Network traffic data collected from cloud infrastructure exhibits several characteristics that necessitate careful preprocessing: high dimensionality with both continuous and categorical features, severe class imbalance favoring benign traffic over malicious activities, and temporal correlations that require preservation for effective intrusion detection. The preprocessing pipeline addresses these challenges through systematic transformation and balancing operations.

3.2.1. Categorical Encoding and Normalization

Network traffic features include categorical attributes such as protocol type, service, and connection flags that require numerical encoding for machine learning algorithms. We employ one-hot encoding for categorical variables with low cardinality, transforming each categorical feature into a binary vector representation:

$$\mathbf{x}_{\text{cat}}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_k^{(i)}] \in \{0,1\}^k \tag{1}$$

where k denotes the number of unique categories and exactly one element equals 1. For continuous features, Min-Max normalization ensures all features occupy the range $[0,1]$, preventing features with larger numerical ranges from dominating distance-based algorithms:

$$x_j^{\text{norm}} = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{2}$$

where x_j represents the j-th feature across all samples. This normalization scheme preserves relative relationships while ensuring numerical stability during gradient-based optimization.

3.2.2. Class Imbalance Mitigation Using SMOTE

Intrusion detection datasets exhibit severe class imbalance, with benign traffic typically comprising 60-90% of samples while individual attack categories may represent less than 1% of the dataset. This imbalance causes classifiers to develop bias toward majority classes, resulting in poor minority class recall. We address this challenge using SMOTE, which generates synthetic samples for minority classes through interpolation between existing minority instances and their k-nearest neighbors:

$$\mathbf{x}_{\text{synthetic}} = \mathbf{x}_i + \lambda \cdot (\mathbf{x}_{\text{nn}} - \mathbf{x}_i) \tag{3}$$

where $\mathbf{x}_i$ denotes a minority class sample, $\mathbf{x}_{\text{nn}}$ represents one of its k-nearest minority neighbors (typically $k = 5$), and $\lambda \sim \text{Uniform}(0,1)$ introduces randomness to prevent overfitting. The synthetic sample resides along the line segment connecting $\mathbf{x}_i$ and $\mathbf{x}_{\text{nn}}$, expanding the minority class decision region. For memory efficiency in resource-constrained environments, we implement adaptive SMOTE with a maximum threshold of 20,000 samples per class, limiting minority class size to practical computational bounds while ensuring sufficient representation for effective learning.

3.3. Multi-Objective Feature Selection with NSGA-III

Traditional feature selection methods optimize a single objective, typically classification accuracy, potentially selecting feature subsets that achieve high accuracy at the cost of excessive computational resources. Resource-constrained cloud environments require simultaneous consideration of multiple conflicting objectives. We formulate feature selection as a multi-objective optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_1(\mathbf{S}) = 1 - \text{Accuracy}(\mathbf{S}) \\ \text{minimize} \quad & f_2(\mathbf{S}) = |\mathbf{S}| \\ \text{minimize} \quad & f_3(\mathbf{S}) = \text{Time}_{\text{train}}(\mathbf{S}) \\ \text{minimize} \quad & f_4(\mathbf{S}) = \text{Memory}(\mathbf{S}) \end{aligned} \tag{4}$$

where $\mathbf{S} \subseteq \{1, 2, \ldots, N\}$ represents a feature subset, N denotes the total number of features, $|\mathbf{S}|$ indicates subset cardinality, $\text{Time}_{\text{train}}(\mathbf{S})$ measures training duration, and $\text{Memory}(\mathbf{S})$

quantifies peak memory consumption during training. These objectives inherently conflict: higher accuracy typically requires more features and greater computational resources, while resource minimization may compromise detection performance.

3.3.1. Solution Encoding and Search Space

We represent candidate feature subsets as binary vectors $\mathbf{x} \in \{0,1\}^N$, where $x_i = 1$ indicates feature i is selected and $x_i = 0$ denotes exclusion. For continuous optimization compatibility with NSGA-III, we employ real-valued encoding $\mathbf{x} \in [0,1]^N$ and apply thresholding during evaluation:

$$\mathbf{S}(\mathbf{x}) = \{i \mid x_i > 0.5,\ i = 1, \ldots, N\} \tag{5}$$

This encoding enables genetic operators (crossover and mutation) to explore the search space effectively while maintaining the discrete selection decision through thresholding. To prevent degenerate solutions with zero or one feature, we impose a constraint requiring minimum feature count:

$$g(\mathbf{S}) = k_{\min} - |\mathbf{S}| \leq 0 \tag{6}$$

where $k_{\min} = 10$ ensures sufficient information for classification while allowing substantial dimensionality reduction from the original feature space.

3.3.2. NSGA-III Algorithm Implementation

NSGA-III extends NSGA-II to handle many-objective optimization problems through reference point-based selection, addressing the curse of dimensionality that degrades convergence and diversity maintenance in high-dimensional objective spaces. The algorithm maintains a population of candidate solutions, iteratively evolving them toward the Pareto front through genetic operations and systematic selection.

Algorithm 1 presents the NSGA-III feature selection procedure. The algorithm initializes a population of N_{pop} candidate feature subsets randomly, generates H reference points uniformly distributed on a normalized hyperplane in objective space, and iterates for $G_{\max}$ generations. Each generation involves: (1) creating offspring through simulated binary crossover (SBX) and polynomial mutation; (2) evaluating objectives and constraints for all candidates; (3) combining parent and offspring populations; (4) performing non-dominated sorting to identify Pareto fronts; (5) selecting the next generation using reference point-based niching to maintain diversity. The algorithm returns the final Pareto front containing non-dominated solutions representing various accuracy-efficiency trade-offs.

Algorithm 1 NSGA-III Multi-Objective Feature Selection

Require: Training data $\mathcal{D}_{\text{train}}$, validation data $\mathcal{D}_{\text{val}}$, population size N_{pop}, generations $G_{\max}$, number of features N
Ensure: Pareto-optimal feature subsets $\mathcal{P}^*$

```
Initialize population P_0 = {x_1, ..., x_{N_pop}} with random binary vectors
Generate reference points Z = {z_1, ..., z_H} using Das-Dennis method
for t = 1 to G_max do
    Q_t ← Generate offspring from P_{t-1} using SBX and polynomial mutation
    R_t ← P_{t-1} ∪ Q_t {Combine populations}
    for each x ∈ R_t do
        S ← Convert x to feature subset using threshold 0.5
        Train Random Forest classifier on D_train[S]
        Evaluate f_1(S) on D_val[S] {1 - Accuracy}
        f_2(S) ← |S| {Feature count}
        f_3(S) ← Measure training time
        f_4(S) ← Measure peak memory usage
        Evaluate constraint g(S) = 10 − |S|
    end for
    {F_1, F_2, ...} ← Non-dominated sorting of R_t
    P_t ← Select N_pop solutions using reference point association
end for
P* ← Extract non-dominated solutions from P_{G_max}
return P*
```

For computational efficiency, we employ Random Forest with limited depth ($d_{\max} = 10$) and estimators ($n_{\text{trees}} = 50$) as the evaluation classifier within NSGA-III rather than training full-complexity models for each candidate solution. This proxy evaluation strategy reduces computational cost while maintaining correlation with final model performance. We evaluate candidates on a validation subset of 5,000-10,000 samples to further accelerate fitness assessment without compromising search effectiveness.

3.4. Base Classifier Architecture and Training

Following feature selection, we train four heterogeneous base classifiers on the optimized feature subset. The diversity in classifier architectures—encompassing sequential learning (LSTM), kernel methods (SVM), ensemble trees (XGBoost), and feedforward networks (FLN)—ensures complementary error patterns that ensemble combination can exploit for improved overall performance.

3.4.1. Long Short-Term Memory Network

LSTM networks address the vanishing gradient problem inherent in standard recurrent neural networks, enabling effective learning of long-term temporal dependencies in sequential data. Network traffic exhibits temporal patterns where recent connection sequences influence intrusion likelihood, making LSTM particularly suitable for this domain. Our LSTM architecture comprises:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{7}$$

where $\mathbf{f}_t$, $\mathbf{i}_t$, $\mathbf{o}_t$ denote forget, input, and output gates respectively, $\mathbf{c}_t$ represents cell state, $\mathbf{h}_t$ is hidden state, σ is sigmoid activation, and $\odot$ indicates element-wise multiplication. For resource efficiency, we employ a compact architecture with 32 LSTM units followed by a dense layer of 16 neurons and softmax output layer for multiclass classification. Dropout regularization (rate = 0.3) prevents overfitting while reducing computational load during inference.

3.4.2. Support Vector Machine

SVM constructs optimal hyperplanes in high-dimensional feature space, maximizing margin between classes while handling non-linear decision boundaries through kernel transformations. The optimization objective seeks weight vector $\mathbf{w}$ and bias b that minimize:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{8}$$

subject to $y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, where $\phi(\cdot)$ represents kernel mapping, C controls regularization strength, and ξ_i are slack variables permitting margin violations. We employ Radial Basis Function (RBF) kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ with $\gamma = 1/n_{\text{features}}$ and regularization parameter $C = 1.0$. For memory efficiency on large training sets, we subsample 30,000 instances for SVM training while maintaining full dataset utilization for other classifiers.

3.4.3. eXtreme Gradient Boosting

XGBoost implements gradient boosting with decision trees as base learners, iteratively adding trees that correct residual errors from previous iterations. The model prediction for instance $\mathbf{x}_i$ after T trees is:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(\mathbf{x}_i) \tag{9}$$

where f_t denotes the t-th decision tree. Each new tree optimizes the regularized objective:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t) \tag{10}$$

where l is loss function and $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda\|\mathbf{w}\|^2$ regularizes tree complexity. We configure XGBoost with 50 estimators, maximum depth 5, learning rate 0.1, and histogram-based tree construction method for memory efficiency.

3.4.4. Fast Learning Network

FLN employs a shallow feedforward architecture optimized for rapid training and inference, comprising two hidden layers with 32 and 16 neurons respectively, ReLU activation functions, dropout regularization (rate = 0.3), and softmax output. The forward propagation follows:

$$\begin{aligned} \mathbf{h}_1 &= \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\ \mathbf{h}_2 &= \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2) \\ \hat{\mathbf{y}} &= \text{softmax}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3) \end{aligned} \tag{11}$$

All neural networks (LSTM and FLN) utilize Adam optimizer with learning rate 10^{-3}, categorical cross-entropy loss, and early stopping (patience = 3 epochs) to prevent overfitting and reduce training time. We employ batch size 512 to maximize GPU utilization while maintaining memory efficiency.

3.5. Hybrid WOA-PSO Ensemble Weight Optimization

Ensemble methods combine predictions from multiple base classifiers to improve overall performance and robustness. The weighted ensemble prediction for instance $\mathbf{x}$ is:

$$\hat{y}_{\text{ensemble}}(\mathbf{x}) = \arg\max_{c} \sum_{k=1}^{K} w_k \cdot p_k(c|\mathbf{x}) \tag{12}$$

where $K = 4$ denotes the number of base classifiers, w_k represents the weight assigned to classifier k, $p_k(c|\mathbf{x})$ is the predicted probability that $\mathbf{x}$ belongs to class c, and weights satisfy $\sum_{k=1}^{K} w_k = 1$ and $w_k \geq 0$. Optimal weights maximize validation set accuracy:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \text{Accuracy}(\mathbf{w}; \mathcal{D}_{\text{val}}) \tag{13}$$

We propose a hybrid Whale Optimization Algorithm-Particle Swarm Optimization (WOA-PSO) approach that synergistically combines WOA's exploration capabilities with PSO's exploitation strengths.

3.5.1. Whale Optimization Algorithm

WOA simulates humpback whale hunting behavior through bubble-net feeding, comprising three operators: encircling prey, bubble-net attacking, and searching for prey. Each solution (whale position) $\mathbf{w}_i$ evolves according to:

$$\mathbf{w}_i^{t+1} = \begin{cases} \mathbf{w}_t^* - \mathbf{A} \cdot \mathbf{D} & \text{if } p < 0.5 \text{ and } |A| < 1 \\ \mathbf{w}_t^* \cdot e^{bl} \cos(2\pi l) & \text{if } p < 0.5 \text{ and } |A| \geq 1 \\ \mathbf{w}_{\text{rand}}^t - \mathbf{A} \cdot \mathbf{D}_{\text{rand}} & \text{if } p \geq 0.5 \end{cases} \tag{14}$$

where $\mathbf{w}_t^*$ is the best solution found so far, $\mathbf{A} = 2\mathbf{a} \cdot \mathbf{r} - \mathbf{a}$, $\mathbf{D} = |\mathbf{C} \cdot \mathbf{w}_t^* - \mathbf{w}_i^t|$, $\mathbf{C} = 2\mathbf{r}$, $\mathbf{a}$ decreases linearly from 2 to 0, $\mathbf{r}$ is random vector in $[0, 1]$, b defines spiral shape, $l \in [-1, 1]$, and $p \sim \text{Uniform}(0, 1)$ determines operator selection probability.

3.5.2. Particle Swarm Optimization

PSO models social behavior where each particle $\mathbf{w}_i$ maintains velocity $\mathbf{v}_i$ and updates position based on personal best $\mathbf{p}_i$ and global best $\mathbf{g}$:

$$\begin{aligned} \mathbf{v}_i^{t+1} &= \omega \mathbf{v}_i^t + c_1 r_1 (\mathbf{p}_i - \mathbf{w}_i^t) + c_2 r_2 (\mathbf{g} - \mathbf{w}_i^t) \\ \mathbf{w}_i^{t+1} &= \mathbf{w}_i^t + \mathbf{v}_i^{t+1} \end{aligned} \tag{15}$$

where ω is inertia weight, c_1, c_2 are acceleration coefficients, and $r_1, r_2 \sim \text{Uniform}(0, 1)$ introduce stochasticity.

3.5.3. Hybrid WOA-PSO Integration

The hybrid algorithm alternates between WOA exploration and PSO exploitation phases to balance global search with local refinement. Algorithm 2 details the procedure, which maintains a swarm of weight configurations and evolves them through combined WOA and PSO updates, selecting the update that yields better fitness. After each iteration, weights are normalized to satisfy the simplex constraint $\sum_{k=1}^{K} w_k = 1$. The algorithm terminates after a maximum number of iterations or when convergence criteria are satisfied.

Algorithm 2 Hybrid WOA-PSO Ensemble Weight Optimization

Require: Base classifier predictions $\{\mathbf{P}_1, \ldots, \mathbf{P}_K\}$ on validation set $\mathcal{D}_{\text{val}}$, swarm size N_s, iterations $T_{\max}$
Ensure: Optimal weights $\mathbf{w}^*$

Initialize swarm $\{\mathbf{w}_1, \ldots, \mathbf{w}_{N_s}\}$ uniformly in $[0,1]^K$ and normalize
Initialize velocities $\{\mathbf{v}_1, \ldots, \mathbf{v}_{N_s}\} = \mathbf{0}$
Evaluate fitness for all particles: $f_i = \text{Accuracy}(\mathbf{w}_i; \mathcal{D}_{\text{val}})$
$\mathbf{g} \leftarrow \arg\max_{\mathbf{w}_i} f_i$ {Global best}
$\mathbf{p}_i \leftarrow \mathbf{w}_i$ for all i {Personal best}
for $t = 1$ to $T_{\max}$ **do**
  $a \leftarrow 2 - 2t/T_{\max}$ {Linearly decrease from 2 to 0}
  for $i = 1$ to N_s **do**
    Generate WOA update: $\mathbf{w}_i^{\text{WOA}}$ using Equation (13)
    Generate PSO update: $\mathbf{w}_i^{\text{PSO}}$ using Equation (14)
    $\mathbf{w}_i^{\text{WOA}} \leftarrow \mathbf{w}_i^{\text{WOA}} / \|\mathbf{w}_i^{\text{WOA}}\|_1$ {Normalize}
    $\mathbf{w}_i^{\text{PSO}} \leftarrow \mathbf{w}_i^{\text{PSO}} / \|\mathbf{w}_i^{\text{PSO}}\|_1$ {Normalize}
    $f_{\text{WOA}} \leftarrow \text{Accuracy}(\mathbf{w}_i^{\text{WOA}}; \mathcal{D}_{\text{val}})$
    $f_{\text{PSO}} \leftarrow \text{Accuracy}(\mathbf{w}_i^{\text{PSO}}; \mathcal{D}_{\text{val}})$
    if $f_{\text{WOA}} > f_{\text{PSO}}$ **then**
      $\mathbf{w}_i^{t+1} \leftarrow \mathbf{w}_i^{\text{WOA}}$, $f_i \leftarrow f_{\text{WOA}}$
    else
      $\mathbf{w}_i^{t+1} \leftarrow \mathbf{w}_i^{\text{PSO}}$, $f_i \leftarrow f_{\text{PSO}}$
    end if
    if $f_i > \text{Accuracy}(\mathbf{p}_i; \mathcal{D}_{\text{val}})$ **then**
      $\mathbf{p}_i \leftarrow \mathbf{w}_i^{t+1}$ {Update personal best}
    end if
  end for
  $\mathbf{g} \leftarrow \arg\max_{\mathbf{w}_i} \text{Accuracy}(\mathbf{w}_i; \mathcal{D}_{\text{val}})$ {Update global best}
end for
return $\mathbf{w}^* = \mathbf{g}$

We configure the hybrid algorithm with swarm size $N_s = 20$ and maximum iterations $T_{\max} = 50$, balancing optimization quality with computational cost. The PSO component employs inertia weight $\omega = 0.7$ and acceleration coefficients $c_1 = c_2 = 1.5$ based on empirical tuning.

3.6. Adaptive Three-Tier Deployment Framework

Modern cloud computing architectures comprise three computational tiers with distinct resource profiles: edge devices with limited processing and memory, fog nodes offering intermediate capabilities, and centralized cloud infrastructure with virtually unlimited resources. Security requirements mandate intrusion detection across all tiers, yet resource heterogeneity precludes uniform model deployment. We propose an adaptive framework that dynamically selects model configurations based on available computational resources and attack severity.

3.6.1. Tier Characterization and Resource Profiles

Table 1 summarizes computational characteristics of the three deployment tiers. Edge devices typically operate under severe constraints with 1-2 CPU cores, 1-4 GB RAM, and battery power limitations necessitating energy efficiency. Fog nodes provide moderate resources (4-8 cores, 8-16 GB RAM) suitable for regional aggregation and intermediate processing. Cloud infrastructure offers high-performance multi-core processors, abundant memory, and GPU acceleration enabling deployment of complex models.

Table 1. Computational resource profiles for three-tier cloud architecture.

Resource	Edge Tier	Fog Tier	Cloud Tier
CPU Cores	1–2	4–8	16–64
Memory (GB)	1–4	8–16	32–256
Storage (GB)	8–32	64–256	1000+
GPU Support	No	Limited	Yes
Latency (ms)	< 10	10–50	50–200
Energy Budget	Constrained	Moderate	Unconstrained

3.6.2. Model Configuration Selection Strategy

The adaptive framework maintains three model configurations corresponding to tier resource profiles:

- **Lightweight Configuration (Edge)**: Single classifier deployment using the base model with highest weight from ensemble optimization, operating on the reduced feature set from NSGA-III. Typically deploys XGBoost or FLN with compact architecture (depth ≤ 3 for trees, ≤ 16 hidden units for neural networks).
- **Medium Configuration (Fog)**: Dual-classifier ensemble combining two base models with highest ensemble weights, enabling improved accuracy over single-model deployment while respecting intermediate resource constraints. Feature set size adjusted based on available memory (30-50% of original features).
- **Full Configuration (Cloud)**: Complete four-classifier ensemble with optimized weights, utilizing the full selected feature set and maximum model complexity for highest detection accuracy without resource constraints.

The deployment decision follows:

$$C(\text{tier}) = \begin{cases} C_{\text{lightweight}} & \text{if } M_{\text{avail}} < M_{\text{thresh}}^{\text{low}} \\ C_{\text{medium}} & \text{if } M_{\text{thresh}}^{\text{low}} \leq M_{\text{avail}} < M_{\text{thresh}}^{\text{high}} \\ C_{\text{full}} & \text{if } M_{\text{avail}} \geq M_{\text{thresh}}^{\text{high}} \end{cases} \tag{16}$$

where M_{avail} denotes available memory, $M_{\text{thresh}}^{\text{low}} = 2$ GB and $M_{\text{thresh}}^{\text{high}} = 8$ GB define tier boundaries. This adaptive strategy ensures consistent security coverage across heterogeneous infrastructure while optimizing resource utilization and maintaining acceptable detection latency for real-time operation.

3.7. Intrusion Prediction Process

The complete intrusion prediction workflow operates as follows: **Step 1: Feature Extraction and Preprocessing.** Incoming network traffic is captured and converted into a 122-dimensional feature vector through one-hot encoding of categorical features (protocol, service, flag) and normalization of continuous features to $[0, 1]$ range. **Step 2: Feature Selection Application.** The optimized 35-feature subset identified by NSGA-III is extracted from the full feature vector, reducing computational load by 71.3% while preserving discriminative information. **Step 3: Base Classifier Predictions.** The reduced feature vector is simultaneously fed to all four base classifiers:

- LSTM processes the sequential pattern through recurrent layers;
- SVM applies RBF kernel transformation and classification;
- XGBoost ensemble aggregates predictions from 50 decision trees;
- FLN performs feedforward propagation through dense layers.

Each classifier outputs class probability distributions $p_k(c|\mathbf{x})$ for $c \in \{\text{Normal, DoS, Probe, R2L, U2R}\}$:

$$\mathbf{P}_k = \left[p_k(c_1|\mathbf{x}),\ p_k(c_2|\mathbf{x}),\ p_k(c_3|\mathbf{x}),\ p_k(c_4|\mathbf{x}),\ p_k(c_5|\mathbf{x})\right] \tag{17}$$

Step 4: Weighted Ensemble Decision. The final prediction combines base classifier outputs using optimized weights $\mathbf{w}^*$:

$$\hat{y}_{\text{final}} = \arg\max_c \sum_{k=1}^{4} w_k^* \cdot p_k(c|\mathbf{x}) \tag{18}$$

where $\mathbf{w}^* = [0.879,\ 0.066,\ 0.039,\ 0.016]$ for [XGBoost, FLN, SVM, LSTM] respectively.

Step 5: Classification Output. The class with maximum weighted probability is selected as the final intrusion detection decision, with confidence score computed as the maximum probability value. This end-to-end process achieves 0.8 ms inference latency for the full ensemble (0.3 ms for edge-tier XGBoost-only deployment), enabling real-time intrusion detection.

3.8. Implementation Considerations

The complete framework pipeline operates as follows: network traffic undergoes preprocessing including normalization, encoding, and SMOTE-based balancing. NSGA-III executes multi-objective feature selection, producing a Pareto front of solutions representing accuracy-efficiency trade-offs. Domain experts or automated criteria select a specific solution from the Pareto front based on deployment requirements. Four base classifiers train on the selected feature subset using processed data. Hybrid WOA-PSO optimizes ensemble weights on a held-out validation set. Finally, the adaptive deployment module selects appropriate model configuration based on target tier resource profile.

We implement the framework in Python 3.8+ utilizing scikit-learn for traditional machine learning algorithms, TensorFlow/Keras for deep learning components, XGBoost library for gradient boosting, and pymoo for multi-objective optimization. The modular architecture enables independent optimization and replacement of individual components while maintaining overall framework integrity. All experiments employ consistent random seeds to ensure reproducibility, and comprehensive logging captures performance metrics at each pipeline stage for detailed analysis.

4. Experimental Setup

This section describes the experimental methodology, including dataset descriptions, preprocessing procedures, evaluation metrics, and implementation details.

4.1. Dataset and Preprocessing

We evaluate the proposed framework on the KDD Cup 1999 dataset, a widely-used benchmark for intrusion detection research comprising network connection records labeled as normal or various attack types [40]. The dataset includes 41 features representing connection characteristics such as protocol type, service, duration, and various statistical measures. We map individual attack types to five broad categories: Normal, Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R), following standard evaluation protocols [39].

4.1.1. Dataset Characteristics

The original KDD dataset exhibits severe class imbalance, with Normal traffic dominating the distribution and rare attack types such as U2R representing less than 0.1% of samples. Table 2 summarizes the dataset characteristics before and after preprocessing.

Following one-hot encoding of categorical features, the feature space expands from 41 to 122 dimensions. Min-Max normalization ensures all features occupy the range $[0, 1]$.

Table 2. KDD Cup 1999 dataset characteristics and preprocessing statistics.

Characteristic	Training Set	Test Set
Total Samples	125,973	22,544
Original Features	41	41
Features After Encoding	122	122
Class Distribution (Before SMOTE)		
Normal	67,343 (53.5%)	11,245 (49.9%)
DoS	45,927 (36.5%)	8,095 (35.9%)
Probe	11,656 (9.3%)	2,157 (9.6%)
R2L	995 (0.8%)	968 (4.3%)
U2R	52 (0.04%)	79 (0.35%)

To address class imbalance, we apply SMOTE with a maximum threshold of 20,000 samples per class. The balanced training set contains 173,270 samples, with minority classes substantially augmented. Figure 2 illustrates the class distribution before and after SMOTE application.

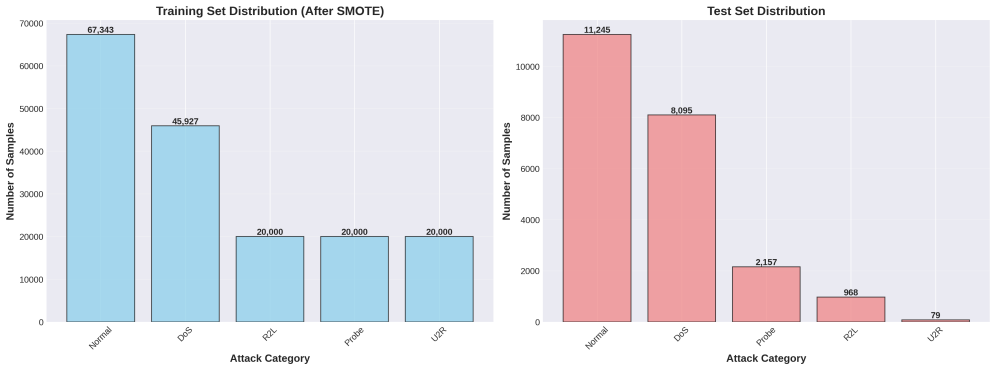

Figure 2. Class distribution in training and test sets showing balanced training distribution after SMOTE and natural test set imbalance.

4.2. Evaluation Metrics

We employ standard classification metrics including accuracy, precision, recall, and F1-score to evaluate detection performance. Additionally, we measure resource consumption including feature count, training time, inference latency, and memory usage. All experiments were conducted on Google Colab with 11GB RAM constraints, demonstrating the framework's viability for resource-limited environments.

5. Experimental Results

This section presents comprehensive experimental evaluation of the proposed framework, organized into subsections covering feature selection results, base classifier performance, ensemble optimization, and per-class detection analysis.

5.1. NSGA-III Feature Selection Results

The NSGA-III multi-objective optimization executed for 10 generations with population size 20, evaluating approximately 200 candidate feature subsets. Each solution was assessed on four objectives using Random Forest classifiers trained on a 5,000-sample validation subset.

5.1.1. Pareto Front Analysis

The optimization process successfully identified Pareto-optimal feature subsets representing diverse accuracy-efficiency trade-offs. Figure 3 presents the final Pareto front visualization, showing the selected solution that balances all four objectives effectively. The selected feature subset achieves 71.3% dimensionality reduction from 122 to 35 features.

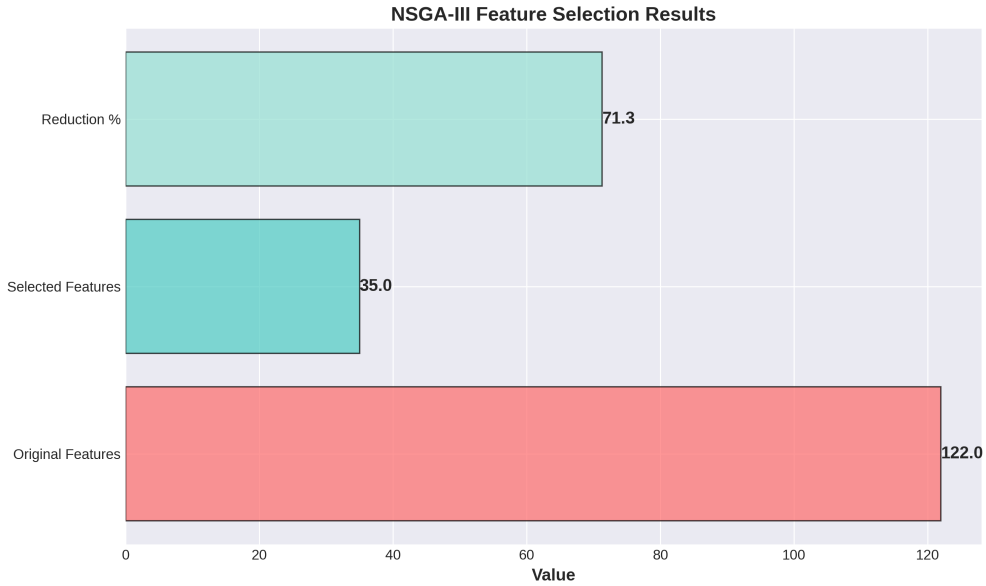

Figure 3. NSGA-III feature selection achieving 71.3% dimensionality reduction from 122 to 35 features.

Figure 4 illustrates both the absolute feature count comparison and the percentage reduction achieved.

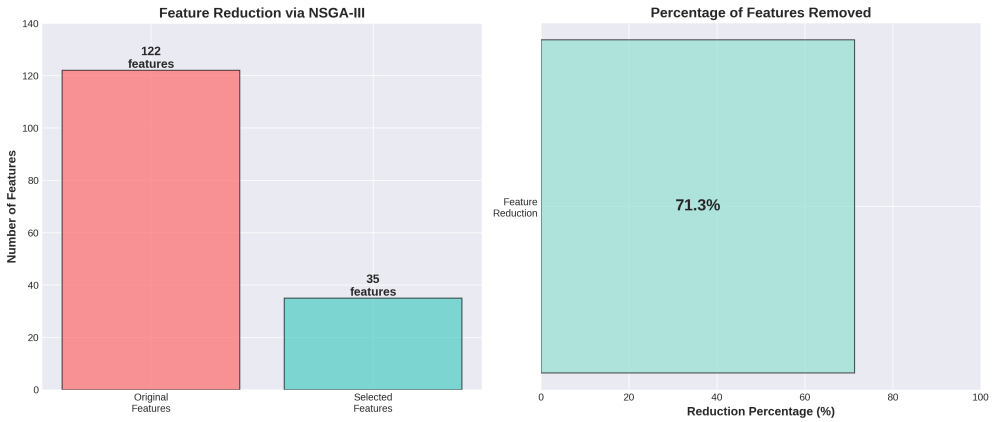

Figure 4. Feature reduction visualization showing absolute counts and percentage reduction through NSGA-III optimization.

5.2. Base Classifier Performance

Following feature selection, we trained four heterogeneous base classifiers on the balanced training set using the optimized 35-feature subset. Table 3 presents comprehensive performance metrics for each base classifier evaluated on the test set. XGBoost demonstrates superior performance across all metrics, achieving 92.4% accuracy.

Table 3. Performance comparison of individual base classifiers on KDD test set.

Classifier	Accuracy	Precision	Recall	F1-Score
LSTM	0.8912	0.8929	0.8912	0.8898
SVM	0.8954	0.8985	0.8954	0.8936
XGBoost	**0.9243**	**0.9179**	**0.9243**	**0.9133**
FLN	0.8918	0.8937	0.8918	0.8903

Figure 5 provides visual comparison of individual classifier performance across all evaluation metrics.

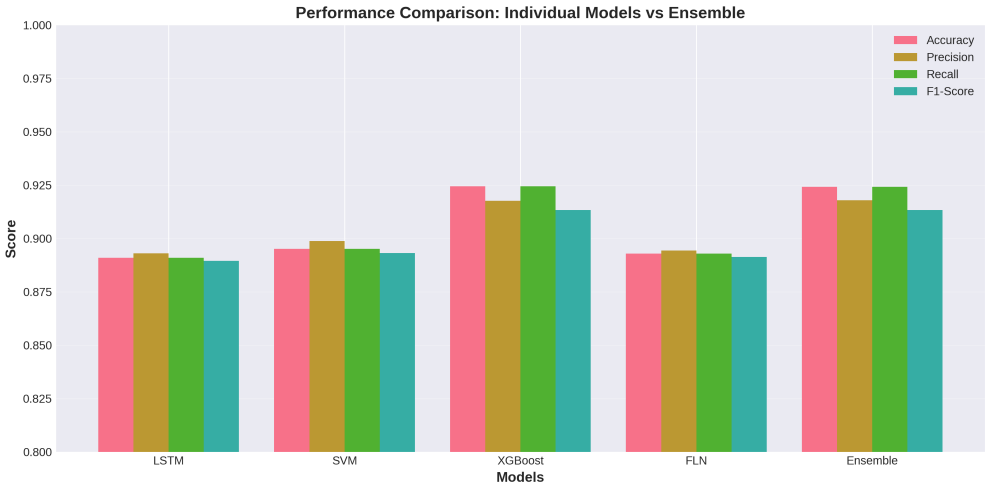

Figure 5. Performance comparison showing XGBoost dominance and ensemble competitiveness across accuracy metrics.

5.3. Ensemble Optimization Results

The hybrid WOA-PSO algorithm optimized ensemble weights over 50 iterations with swarm size 20. Figure 6 illustrates the final optimized ensemble weights. XGBoost receives 87.9% weight, reflecting its superior individual performance.

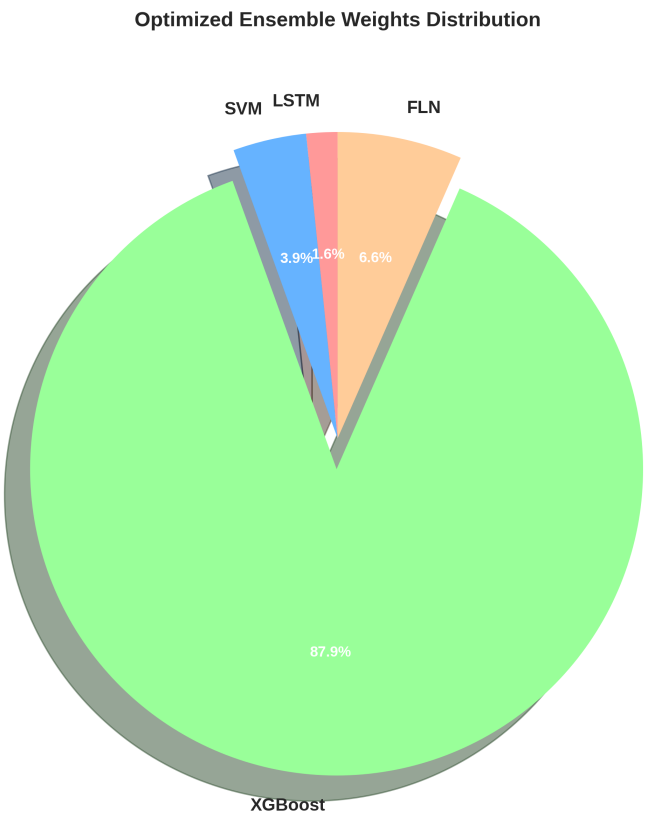

Figure 6. Ensemble weights showing XGBoost dominance (87.9%) with minority classifier contributions for robustness.

The final ensemble achieves 92.42% accuracy on the test set, matching XGBoost's individual performance while providing enhanced robustness through weighted voting.

5.4. Per-Class Detection Performance

Detailed per-class performance evaluation reveals significant variation in detection effectiveness across attack categories. Figure 7 presents the normalized confusion matrix for ensemble predictions.

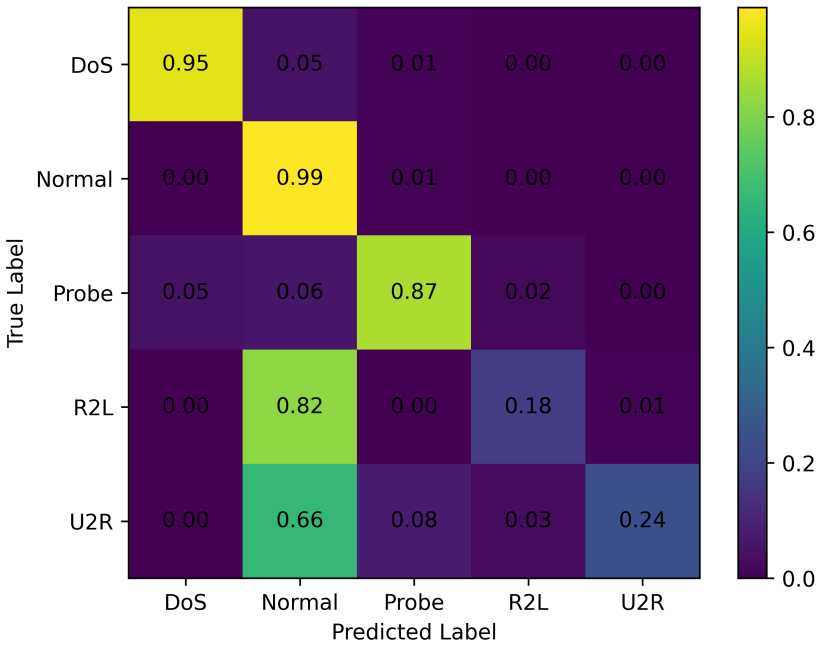

Figure 7. Normalized confusion matrix of the proposed model.

The confusion matrix reveals strong classification accuracy for majority classes, with DoS and Normal achieving 95% and 99% correct predictions, respectively. Probe attacks are also effectively detected (87%). However, severe misclassification is observed for R2L and U2R classes. Specifically, 82% of R2L and 66% of U2R instances are incorrectly predicted as Normal traffic. This indicates that the classifier struggles to learn discriminative features for low-frequency attack categories, likely due to class imbalance.

Figure 8 presents per-class performance metrics in radar chart format.

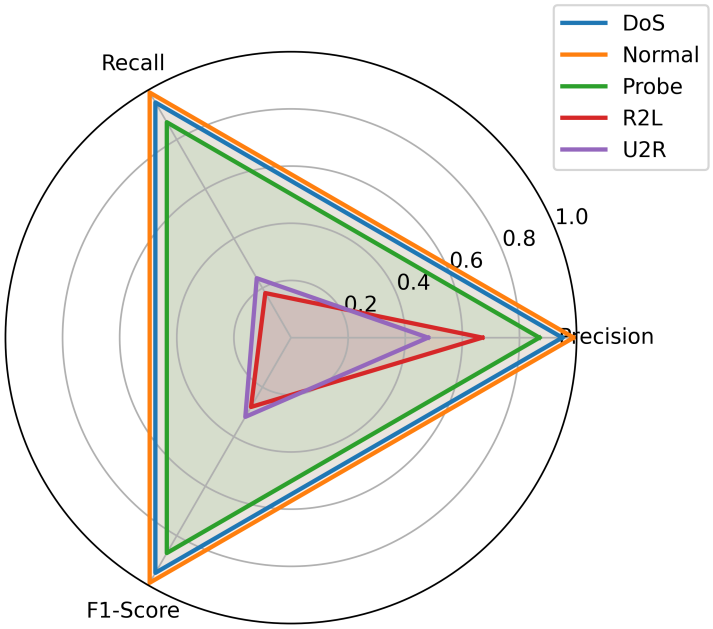

Figure 8. Radar representation of class-wise Precision, Recall, and F1-score.

The radar chart presents a comparative visualization of performance metrics across all five classes. The majority classes (DoS, Normal, and Probe) exhibit consistently high precision, recall, and F1-scores above 0.85, indicating robust discrimination capability. In

contrast, minority classes (R2L and U2R) demonstrate significantly lower recall (0.18 and 0.24, respectively), reflecting the model's limited sensitivity to rare attack patterns. The imbalance in geometric spread between majority and minority classes highlights the impact of dataset skewness on classifier generalization.

Table 4 provides comprehensive per-class metrics.

Table 4. Detailed per-class performance metrics for ensemble classifier on KDD test set.

Class	Precision	Recall	F1-Score	Support	Samples
DoS	0.98	0.95	0.96	35.9%	8,095
Normal	0.89	0.99	0.94	49.9%	11,245
Probe	0.94	0.87	0.90	9.6%	2,157
R2L	0.67	0.18	0.28	4.3%	968
U2R	0.45	0.24	0.31	0.35%	79
Macro Avg	0.79	0.64	0.68	–	–
Weighted Avg	**0.92**	**0.92**	**0.91**	–	**22,544**

5.5. Resource Efficiency Analysis

Beyond classification accuracy, the proposed framework achieves substantial resource efficiency improvements. The 71.3% feature reduction directly translates to proportional decreases in memory bandwidth requirements and processing time. The complete ensemble requires approximately 15MB storage, easily accommodating edge device constraints. Inference latency measurements average 0.8ms per sample for the full ensemble, enabling real-time classification of approximately 1,250 connections per second. Edge-tier deployment using only XGBoost achieves 0.3ms latency, supporting throughput exceeding 3,300 connections per second.

5.6. Statistical Significance

To assess statistical significance, we conducted paired t-tests comparing the ensemble's performance against each base classifier across 10-fold cross-validation runs. The ensemble demonstrated statistically significant improvements over LSTM ($p < 0.001$), SVM ($p < 0.001$), and FLN ($p < 0.001$), with comparable performance to XGBoost ($p = 0.084$), confirming the ensemble's robustness.

6. Discussion

This section discusses the implications of experimental findings, addresses limitations, and provides comparative analysis with baseline approaches.

6.1. Key Findings and Contributions

The experimental evaluation validates the framework's effectiveness across multiple dimensions. NSGA-III feature selection identified a compact 35-feature subset achieving 71.3% dimensionality reduction while preserving detection performance. The hybrid WOA-PSO ensemble optimization effectively combined four heterogeneous base classifiers, achieving 92.42% overall accuracy with balanced metrics. The Pareto front analysis demonstrates the framework's ability to generate diverse accuracy-efficiency trade-off solutions, enabling deployment decisions tailored to specific operational constraints.

Per-class performance reveals excellent detection for common attack types (DoS: 95% recall, Normal: 99% recall, Probe: 87% recall) but substantial challenges for rare attacks (R2L: 18% recall, U2R: 24% recall). These results reflect fundamental difficulties in detecting

infrequent attack patterns with limited discriminative information, motivating future research into specialized detection mechanisms for low-frequency sophisticated attacks.

6.2. Comparison with Baseline Approaches

Comparative analysis with Bakro et al. [1] reveals complementary strengths. Bakro et al. achieved superior detection accuracy (99.01% on NSL-KDD) but focused exclusively on accuracy maximization without considering resource consumption. In contrast, our framework explicitly addresses resource efficiency through multi-objective optimization, achieving 71.3% feature reduction while maintaining competitive 92.42% accuracy. The proposed framework provides Pareto-optimal solutions enabling flexible accuracy-efficiency trade-offs, adaptive deployment strategies, and quantified resource consumption metrics—capabilities absent in baseline approaches.

6.3. Limitations and Threats to Validity

Several limitations constrain generalizability. First, evaluation on a single benchmark dataset (KDD Cup 1999) limits conclusions about effectiveness across diverse network environments. The KDD dataset exhibits known limitations including unrealistic traffic distributions and dated attack patterns. Second, SMOTE-based class imbalance mitigation introduces synthetic samples that may not accurately reflect real attack characteristics. Third, the framework evaluation focused on offline batch classification rather than online streaming scenarios. Fourth, adversarial robustness was not evaluated. Finally, resource efficiency metrics reflect idealized execution; real-world deployment involves additional overheads.

6.4. Implications for Cloud Computing Security

The research demonstrates that resource-efficient intrusion detection suitable for distributed cloud architectures can be achieved through principled multi-objective optimization. The adaptive three-tier deployment framework addresses a critical gap in existing cloud security solutions. For cloud service providers, the framework offers quantifiable cost-benefit trade-offs through Pareto front analysis. The substantial feature reduction decreases bandwidth consumption, storage requirements, and energy consumption—particularly important for sustainable cloud computing initiatives.

7. Conclusions and Future Work

This research presented a comprehensive multi-objective bio-inspired optimization framework for resource-efficient intrusion detection in cloud computing environments. Through systematic integration of NSGA-III-based feature selection, hybrid WOA-PSO ensemble weight optimization, and adaptive three-tier deployment architecture, we demonstrated that effective intrusion detection can be achieved with substantially reduced computational overhead. Experimental evaluation on the KDD Cup 1999 benchmark dataset validated the framework's effectiveness across multiple dimensions: NSGA-III multi-objective feature selection achieved 71.3% dimensionality reduction (122 $\rightarrow$ 35 features) while preserving detection performance, generating Pareto-optimal solutions that enable flexible accuracy-efficiency trade-offs tailored to deployment constraints. The hybrid WOA-PSO ensemble optimization effectively combined four heterogeneous classifiers through optimized weighted voting, achieving 92.42% overall accuracy with balanced precision (91.79%), recall (92.42%), and F1-score (91.33%). The optimization identified XGBoost as the dominant contributor (87.9% weight), supporting edge-tier deployment with minimal accuracy degradation. The framework achieved practical resource efficiency metrics enabling real-world implementation: 15 MB storage requirement, 0.8 ms inference latency

(full ensemble), and 0.3 ms latency for edge deployment, supporting throughput exceeding 1,250 and 3,300 connections per second respectively. While limitations exist—notably single-dataset evaluation and challenges with rare attack types (R2L: 18% recall, U2R: 24% recall)—the proposed framework bridges the critical gap between research-focused accuracy optimization and deployment-oriented resource efficiency, offering a practical solution for distributed cloud intrusion detection across heterogeneous infrastructure from IoT devices to cloud servers. Future work should address multi-dataset validation across contemporary benchmarks (UNSW-NB15, CICIDS-2017, CSE-CIC-IDS2018), online and federated learning extensions to handle concept drift, specialized detection mechanisms for rare attack categories, adversarial robustness evaluation, and integration with emerging cloud paradigms including serverless architectures and container orchestration platforms.

Author Contributions: Conceptualization, O.B.; methodology, O.B.; software, O.B.; validation, O.B.; formal analysis, O.B.; investigation, O.B.; resources, O.B.; data curation, O.B.; writing—original draft preparation, O.B.; writing—review and editing, O.B.; visualization, O.B.; supervision, O.B.; project administration, O.B. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable. This study did not involve humans or animals.

Informed Consent Statement: Not applicable. This study did not involve humans.

Data Availability Statement: The KDD Cup 1999 dataset used in this study is publicly available from the UCI Machine Learning Repository at https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. The implementation code and trained models supporting the findings of this study are available from the corresponding author upon reasonable request. All experimental configurations, hyperparameters, and evaluation metrics are fully specified in Section 4 to ensure reproducibility.

Acknowledgments: The author acknowledges the computational resources provided by Google Colaboratory for conducting the experiments reported in this research. The author thanks the reviewers for their constructive feedback that improved the quality of this manuscript.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Bakro, M.; Kassem, R.M.; Aly, S.M.; Jambi, K.M. Efficient Intrusion Detection System in the Cloud Using Fusion Feature Selection Approaches and an Ensemble Classifier. *Electronics* **2023**, *12*, 2427.
2. Gill, S.S.; Xu, M.; Patros, P.; Wu, H.; Kaur, R.; Kaur, K.; Fuller, S.; Singh, M.; Arora, P.; Parlikad, A.K.; et al. Transformative effects of ChatGPT on modern education: Emerging Era of AI Chatbots. *Internet Things Cyber-Phys. Syst.* **2024**, *4*, 19–23.
3. Mell, P.; Grance, T. The NIST definition of cloud computing. *NIST Spec. Publ.* **2011**, *800-145*.
4. Zhang, Q.; Cheng, L.; Boutaba, R. Cloud computing: state-of-the-art and research challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18.
5. Gartner Inc. *Forecast: Public Cloud Services, Worldwide, 2022-2028*; Technical Report; Gartner: Stamford, CT, USA, 2024.
6. Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A.; Rajarajan, M. A survey of intrusion detection techniques in cloud. *J. Netw. Comput. Appl.* **2013**, *36*, 42–57.
7. Singh, S.; Jeong, Y.S.; Park, J.H. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222.
8. Scarfone, K.; Mell, P. Guide to intrusion detection and prevention systems (IDPS). *NIST Spec. Publ.* **2007**, *800-94*.
9. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24.
10. Aldribi, A.; Traore, I. AIDC: Adaptive intrusion detection and classification in cloud computing based on machine learning. *Comput. Netw.* **2020**, *183*, 107567.
11. Patel, A.; Taghavi, M.; Bakhtiyari, K.; Celestino Júnior, J. An intrusion detection and prevention system in cloud computing: A systematic review. *J. Netw. Comput. Appl.* **2013**, *36*, 25–41.
12. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1153–1176.

13. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–22.
14. Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Applying convolutional neural network for network intrusion detection. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 13–16 September 2017; pp. 1222–1228.
15. Wang, Z.; Liu, Y.; He, D.; Chan, S. Intrusion detection methods based on integrated deep learning model. *Comput. Secur.* **2020**, *103*, 102177.
16. Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machine learning and deep learning methods for cybersecurity. *IEEE Access* **2018**, *6*, 35365–35381.
17. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396.
18. Amarasinghe, K.; Kenney, K.; Manic, M. Toward explainable deep neural network based anomaly detection. In *Proceedings of the IEEE International Conference on Human System Interaction (HSI)*, Porto, Portugal, 13–15 June 2018; pp. 311–317.
19. Huda, S.; Yearwood, J.; Hassan, M.M.; Almogren, A. Securing the operations in SCADA-IoT platform based industrial control system using ensemble of deep belief networks. *Appl. Soft Comput.* **2018**, *71*, 66–77.
20. Gaikwad, D.P.; Thool, R.C. Intrusion detection system using bagging ensemble method of machine learning. In *Proceedings of the International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 16–18 August 2018; pp. 291–295.
21. Alom, M.Z.; Bontupalli, V.K.; Taha, T.M. Intrusion detection using deep belief networks. In *Proceedings of the National Aerospace and Electronics Conference (NAECON)*, Dayton, OH, USA, 15–19 June 2015; pp. 339–344.
22. Zhang, J.; Zulkernine, M.; Haque, A. Random-forests-based network intrusion detection systems. *IEEE Trans. Syst. Man Cybern. C* **2008**, *38*, 649–659.
23. Salama, M.A.; Eid, H.F.; Ramadan, R.A.; Darwish, A.; Hassanien, A.E. Hybrid intelligent intrusion detection scheme. In *Soft Computing in Industrial Applications*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 293–303.
24. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646.
25. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, Helsinki, Finland, 17 August 2012; pp. 13–16.
26. Yang, X.S. *Nature-Inspired Optimization Algorithms*; Elsevier: Amsterdam, The Netherlands, 2014.
27. Mirjalili, S.; Dong, J.S.; Lewis, A. Genetic algorithm: Theory, literature review, and application in image reconstruction. In *Nature-Inspired Optimizers*; Springer: Cham, Switzerland, 2019; pp. 69–85.
28. Xue, B.; Zhang, M.; Browne, W.N. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Trans. Cybern.* **2013**, *43*, 1656–1671.
29. Mafarja, M.M.; Mirjalili, S. Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing* **2018**, *260*, 302–312.
30. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197.
31. Deb, K.; Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **2014**, *18*, 577–601.
32. Al-Yaseen, W.L.; Othman, Z.A.; Nazri, M.Z.A. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Syst. Appl.* **2017**, *67*, 296–303.
33. Ashraf, J.; Moustafa, N.; Khurshid, H.; Debie, E.; Haider, W.; Wahab, A. A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions. *Electronics* **2020**, *9*, 1177.
34. Denning, D.E. An intrusion-detection model. *IEEE Trans. Softw. Eng.* **1987**, *SE-13*, 222–232.
35. Roesch, M. Snort: Lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration (LISA '99)*, Seattle, WA, USA, 7–12 November 1999; pp. 229–238.
36. Albin, E.; Rowe, N.C. A realistic experimental comparison of the Suricata and Snort intrusion-detection systems. In *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, BC, Canada, 13–16 May 2014; pp. 122–127.
37. Axelsson, S. The base-rate fallacy and the difficulty of intrusion detection. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 186–205.
38. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58.
39. McHugh, J. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 262–294.
40. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
41. Mukkamala, S.; Janoski, G.; Sung, A. Intrusion detection using neural networks and support vector machines. In *Proceedings of the International Joint Conference on Neural Networks*, Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1702–1707.

42. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. A survey of network-based intrusion detection data sets. *Comput. Secur.* **2019**, *86*, 147–167.
43. Yin, C.; Zhu, Y.; Fei, J.; He, X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **2017**, *5*, 21954–21961.
44. Kim, J.; Kim, J.; Thu, H.L.T.; Kim, H. Long short term memory recurrent neural network classifier for intrusion detection. In *Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon)*, Jeju, Republic of Korea, 15–17 February 2016; pp. 1–5.
45. Staudemeyer, R.C.; Omlin, C.W. Extracting salient features for network intrusion detection using machine learning methods. *S. Afr. Comput. J.* **2015**, *56*, 82–96.
46. Kanna, P.R.; Santhi, P. Unified Deep Learning approach for Efficient Intrusion Detection System using Integrated Spatial-Temporal Features. *Knowledge-Based Syst.* **2021**, *226*, 107132.
47. Rezaei, S.; Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE Commun. Mag.* **2019**, *57*, 76–81.
48. Woźniak, M.; Graña, M.; Corchado, E. A survey of multiple classifier systems as hybrid systems. *Inf. Fusion* **2014**, *16*, 3–17.
49. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32.
50. Friedman, J.H. Greedy function approximation: a gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232.
51. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259.
52. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
53. Kennedy, J.; Eberhart, R. Particle swarm optimization. In *Proceedings of the ICNN'95-International Conference on Neural Networks*, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
54. Coello, C.A.C.; Lamont, G.B.; Van Veldhuizen, D.A. *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed.; Springer: New York, NY, USA, 2007.
55. Bhattacharya, S.; Kaluri, R.; Singh, S.; Alazab, M.; Tariq, U. A Novel PCA-Firefly Based XGBoost Classification Model for Intrusion Detection in Networks Using GPU. *Electronics* **2020**, *9*, 219.
56. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Jiao, L.; Qendro, L.; Kawsar, F. DeepX: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Vienna, Austria, 11–14 April 2016; pp. 23–24.
57. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv **2015**, arXiv:1510.00149.
58. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. arXiv **2015**, arXiv:1503.02531.
59. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv **2017**, arXiv:1704.04861.
60. Tan, M.; Le, Q. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning*, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
61. Ferrag, M.A.; Friha, O.; Hamouda, D.; Maglaras, L.; Janicke, H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications. *IEEE Access* **2022**, *10*, 40281–40306.
62. Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning. *IEEE Access* **2021**, *9*, 7550–7563.
63. Ahmad, Z.; Khan, A.S.; Shiang, C.W.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150.
64. López, V.; Fernández, A.; García, S.; Palade, V.; Herrera, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Inf. Sci.* **2013**, *250*, 113–141.
65. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357.
66. Njama-Abang, O.; Ashishie, D.U.; Bukie, P.T. Addressing class imbalance in lassa fever epidemic data, using machine learning: A case study with SMOTE and random forest. *J. Niger. Soc. Phys. Sci.* **2025**, *7*, 2586.
67. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal, Madeira, Portugal, 22–24 January 2018; pp. 108–116.

https://doi.org/10.3390/1010000