IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Dynamic Intrusion Detection Optimization With Swarm Intelligence For Real-Time Adaptive Thresholds And Attack Response

Saurabh Shrivastava[1], Jitendra Singh Kushwah[2] and Pritaj Yadav[3]

[1,3]Department of Computer Science and Engineering, Rabindranath Tagore University, Bhopal, India.
[2]Department of Information Technology, Institute of Technology and Management, Gwalior, India.

Abstract-Intrusion detection systems are very useful in protecting the contemporary networks against all types of cyber attacks which are becoming more complex and dynamic. This paper provides the development of a dynamic intrusion detection system that combines swarm intelligence-based optimization with the latest machine learning and deep learning architectures to allow real-time dynamic thresholds and efficient attack response. A powerful preprocessing pipeline was created, and it includes feature encoding, YeoJohnson transformation, mutual informationbased feature selection and a hybrid approach to class imbalance dealing with SMOTE and ADASYN. Differentiation Evolution was utilized to optimize the important hyper parameters so that global search is made to be efficient and better model generalization is given in case of imbalance. Classical machine learning as well as deep learning models were assessed using the macro-averaged performance measures so as to have a fair evaluation across the traffic classes. The results of experiments show that ensemble and distance-based machine learning models obtained almost perfect detection performance. The 1D Convolutional Neural Network has shown superiority over temporal models, making it a good choice of deep learning techniques, with accuracy of 99.98 percent, precision of 99.99, recall of 99.98 and F1 -score of 99.98. The results prove that swarm intelligence augmented with optimized learning architectures has been able to offer a scalable, robust, and highly accurate means of detecting network intrusions in real-time in the dynamic cybersecurity challenges.

Keyword-Intrusion Detection System, Swarm Intelligence, Differential Evolution, Adaptive Thresholds, Feature Selection and Network Security

1. INTRODUCTION

The high rate of development of digital infrastructure, cloud computing, Internet of Things (IoT), and the 5G-enabled networks has further complicated the nature and size of contemporary cyber environments. Though the developments are facilitating smooth connections and fast data transfer, they are also leaving the systems vulnerable to advanced and chronic cyber attacks. The old security tools like firewalls, signature-based detection and rule-based systems that are static are becoming insufficient to protect against the changing pattern of attacks, zero-day attacks and high volume distributed attacks [1]. In this regard, Intrusion Detection Systems (IDS) have become a very important defense layer with the aim of detecting malicious activities, policy violations and anomalous behaviors in the network and host settings. Traditional IDS methods normally use preset signatures or standardized statistical levels to differentiate between legitimate and unlawful traffic. These systems are highly ineffective in dynamic network environments, and have few adaptability, high false-positive rates and are effective

only against known attacks. The use of static thresholds does not reflect the existing dynamic changes in traffic behavior, and new threats are detected with delays or false identifications. Moreover, the increasing variety of online attacks such as distributed denial-of-service (DDoS) and advanced persistent threats (APTs), insider or malicious and polymorphic malware require smart detection systems that can learn, adapt, and make independent decisions [2].

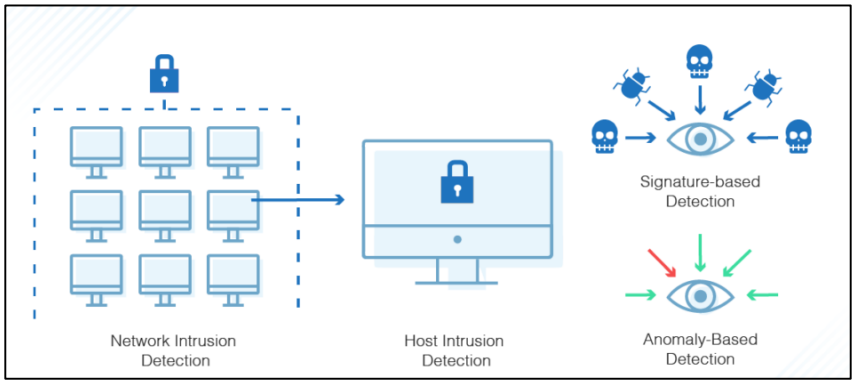

Figure 1 Network Intrusion Detection System

New development and innovations in machine learning and deep learning have made significant contribution to increasing the accuracy of intrusion detection since large data sets can be learnt to develop complexcy. Yet, these data-driven models can be frequently based on predetermined decision boundaries or offline training, which means that they are not as responsive to the rapidly varying attack scenarios. In addition, the dimensionality of feature space, concept drift, and skewed datasets are other factors that complicate the effectiveness of intelligent IDS models. As a way to cope with these shortcomings, the combination of bio-inspired optimization methods with swarm intelligence with IDS is increasingly being considered [3]-[7].

Swarm intelligence is a type of nature-inspired computational methods that simulate the behaviour of self-organizing, decentralized, nature-inspired systems (ant colonies, bird flocks, fish schools, bee swarms, etc.). Some algorithms like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony and Firefly Algorithm (FA) have proven to be exceptionally efficient in solving complex optimization problems in dynamic environment. These algorithms are highly flexible, parallel, and capable of searching the whole world; hence, they can be used in real-time cybersecurity applications where it is necessary to optimally tuning the parameters and respond quickly [8]-[12].

Swarm intelligence provides an effective way to optimize dynamically detection parameters, feature selection, and detection thresholds in the context of intrusion detection. Swarm-based solutions can update detection limits on-the-fly according to real-time traffic dynamics, intensity of threats and system feedback as opposed to using a fixed cutoff value. This dynamic thresholding feature makes its detection much more accurate and the false alarms minimized, especially in highly dynamic network environments. Moreover, swarm intelligence allows collaborative learning between the detection agents so that the parts of the IDS can exchange information and react collectively to a threat as it develops [13].

The other important issue to the current design of IDS is the necessity to respond to attack in real-time. Identifying is not enough without initiating mitigation measures in good time. The more traditional response mechanisms tend to be rule-based and reactive and result in countermeasures that cause delay or ineffectiveness. With swarm intelligence combined with adaptive response measures, IDS designs can prioritize threats, efficiently distribute defensive resources and automatically choose the best response measures including throttling traffic, packet dropping, isolating nodes, or escalating alerts. This leads to closed-loop security system that is not only intrusion detection system but also intelligent and dynamically responsive [14]-[17].

Dynamic Intrusion Detection Optimization with Swarm Intelligence is a new way of looking at security that moves away from static, isolated detection models and toward adaptive, self-optimizing security ecosystems. IDS models can constantly improve their internal settings in response to changes in the environment, the evolution of attacks, and performance feedback by using swarm-based optimization. This method is very useful for real-time applications like smart grids, industrial control systems, cloud

platforms, and IoT networks, where latency limits and operational continuity are very important. This research centers on the creation of a dynamic intrusion detection system that incorporates swarm intelligence methodologies for real-time adaptive threshold adjustment and attack response. The suggested method attempts to improve detection accuracy, reduce false positives, and make sure that cyber threats are dealt with quickly in changing network conditions [18]-[20]. The system aims to tackle significant issues in scalability, flexibility, and resilience by merging the advantages of intelligent learning models with bio-inspired optimization. In the end, this research helps to improve next-generation IDS solutions that can learn on their own, optimize in real time, and defend against advanced cyberattacks before they happen.

2. LITERATURE REVIEW

Ranpara 2025 et.al proposed GreenMU, a new green artificial intelligence–driven intrusion detection framework that aims to improve both energy efficiency and detection performance, which are two of the most important problems in modern IDS design. The framework combines machine learning classifiers like Random Forest and Support Vector Machines with knowledge distillation and adaptive energy-aware optimization to find a compromise between the cost of computing and the accuracy of cybersecurity. The MUGuard algorithm is at the heart of GreenMU. It changes the level of computational complexity in real time based on energy limits and new cyber threats. A lot of testing using the KDD Cup 1999 dataset showed that GreenMU could detect things with an accuracy of almost 99%, which was better than standard IDS models. The framework also cut energy use by 31% and processing time by 15%, making it a great choice for contexts with limited resources, like IoT and edge computing platforms. This study emphasizes the increasing importance of sustainable and energy-efficient IDS solutions; nonetheless, it predominantly concentrates on efficiency optimization and lacks features for adaptive threshold adjustment and intelligent attack response in dynamic network environments [21].

Moriano 2025 et.al performed the inaugural systematic literature review (SLR) on adaptive anomaly detection (AAD) for cyber-physical systems (CPS), tackling the deficiencies of conventional security techniques dependent on static threat classification. The authors scrutinized 397 pertinent studies published from 2013 to November 2023, doing a thorough analysis of 65 essential pieces. A complete taxonomy was created that included attack types, CPS domains, learning paradigms, data handling methodologies, and detection algorithms. The review showed that most current AAD methods focus on either adapting models or processing data quickly, but very few combine both into a single framework. This disjointed approach reveals a substantial research deficiency in the advancement of fully adaptive, real-time intrusion detection systems. The study also pointed to problems with explainability, quantifying uncertainty, and scalability in CPS contexts. The SLR offers significant insights and potential research avenues; yet, it lacks a definitive optimization framework, underscoring the necessity for intelligent, self-optimizing IDS models that can adapt in real-time and respond in a coordinated manner [22].

Mohamed 2025 et.al gave a thorough overview of how artificial intelligence and machine learning are used in cybersecurity, showing how they have changed the way we identify intrusions, classify malware, analyze behavior, and gather threat intelligence. This research focuses on adversarial robustness, investigating methods to enhance AI-based security models against adversarial assaults and data poisoning, in contrast to previous surveys. The authors also looked into federated learning as a way to protect privacy while sharing threat intelligence across decentralized networks. This would allow for real-time cyber security without having to share data in one place. The report also talked about how AI is coming together with new technologies like quantum computing and IoT security. These integrations are seen as key parts of the future generation of adaptive cybersecurity systems. A forward-looking plan was suggested, focusing on AI-driven military architectures that are long-lasting and strong. Even though the study covers a lot of ground, it is mostly theoretical and review-based. It doesn't include any practical information about real-time optimization, adaptive threshold control, or autonomous response mechanisms. This shows that there is a need for practical, optimization-driven IDS frameworks [23].

Popoola 2025 et.al addressed a significant issue in Industrial Internet of Things (IIoT) security: the class imbalance in intrusion detection datasets, by suggesting a multi-stage deep learning (MSDL) method. The research demonstrated that traditional deep learning models frequently do not identify minority-class attacks because of uneven data distributions, resulting in unreported breaches and false alarms in industrial settings. To address this shortcoming, the proposed MSDL framework improves feature learning and classification at several levels, making it easier to find attack types that are not well represented. We tested the suggested strategy on extremely unbalanced datasets, X-IIoT ID and WUSTL-IIoT, and it did better than both baseline deep learning models and the best sampling methods. The results showed that it was much easier to spot rare but important threats, which made IIoT systems more resilient. But the system doesn't include adaptive optimization algorithms for real-time threshold adjustment or attack response coordination, which makes it less useful in industrial networks that are very dynamic and have threats that change over time [24].

Adewale 2024 et.al offered a thorough examination of the use of artificial intelligence into cybersecurity, highlighting its impact on revolutionizing threat identification, vulnerability assessment, and incident response. The research investigated fundamental AI methodologies, including machine learning and natural language processing, and analyzed their utilizations in automated threat prioritization and adaptive risk assessment. The authors showed that intelligent systems are better at dealing with complex and changing cyber threats than standard rule-based security systems by comparing the two. The manuscript also talked on how important AI-powered automation is for speeding up incident response and cutting down on mistakes made by people. There was a lot of talk about the moral and privacy issues that come with using AI, which showed how important it is to be open and make responsible choices. The study provides significant insights into AI's strategic function in cybersecurity; nevertheless, it does not address real-time optimization, swarm-based intelligence, and adaptive IDS frameworks, necessitating more investigation into intelligent, self-optimizing intrusion detection systems [25].

Table 1 Literature Summary

Author / Year	Methodology	Key Findings	Research Gap	Limitations
Qudus (2025) [26]	Conceptual and case-based analysis of AI-driven manufacturing systems using machine learning, predictive analytics, and automation for process control and efficiency improvement.	Demonstrated that AI enhances operational efficiency, predictive maintenance, real-time monitoring, and adaptability in Industry 4.0 environments, leading to improved productivity and sustainability.	Focuses on manufacturing process optimization; lacks discussion on cybersecurity, intrusion detection, or real-time threat adaptation using AI optimization techniques.	Does not address security vulnerabilities of AI-enabled systems; absence of real-time adaptive defense mechanisms and quantitative validation in cyber contexts.
Berguiga et al. (2025) [27]	Hybrid deep learning IDS for IoMT using CNN for feature extraction and LSTM for temporal modeling; deployed in fog computing environment on Raspberry Pi; evaluated on IoTID20 and Edge-IIoTset datasets.	Achieved very high detection performance (Accuracy: 99.92%, F1-score: 99.95%) against multiple DDoS and IoT-based attacks; reduced latency via fog deployment.	Relies on static decision thresholds and fixed model parameters; lacks dynamic optimization and adaptive thresholding under evolving traffic patterns.	High computational complexity; limited explainability; performance may degrade under unseen attack behaviors or real-time concept drift.

Mohale & Obagbuwa (2025) [28]	Systematic review of Explainable AI (XAI) techniques integrated with IDS, focusing on transparency, interpretability, and trustworthiness.	Identified rule-based and tree-based XAI models as more interpretable; highlighted trade-offs between explainability and detection accuracy.	Does not explore optimization techniques such as swarm intelligence for balancing accuracy, explainability, and adaptability in real time.	Mostly qualitative; lacks experimental validation and real-time deployment scenarios; scalability and response automation remain unresolved.
Srinivasan (2024) [29]	Comprehensive review of AI-based security techniques for IoT, including anomaly detection, intrusion prevention, authentication, and threat intelligence.	Concluded that AI significantly improves IoT security resilience against dynamic threats; emphasized future role of intelligent automation.	Does not propose adaptive optimization mechanisms for threshold tuning or coordinated attack response strategies.	Review-based study; absence of implementation details, performance benchmarking, and real-time adaptive evaluation.
Farrukh et al. (2024) [30]	Proposed AIS-NIDS with packet-level analysis and autonomous incremental learning for unknown attack detection using machine learning.	Successfully detected unknown attacks and autonomously learned new attack classes in real-time scenarios, enhancing system adaptability.	Lacks swarm-based optimization for global parameter tuning and does not address adaptive threshold control or coordinated response actions.	Increased processing overhead due to packet-level inspection; scalability concerns in high-speed or large-scale networks.

3. RESEARCH METHODOLOGY

The research methodology delineates the structured processes employed to formulate and assess a resilient intrusion detection framework. Key steps include gathering data from a publicly available network traffic dataset, preprocessing the data to make sure it is of good quality and consistent, exploratory data analysis to find patterns and class imbalance, feature encoding and transformation to make sure the data is represented uniformly, feature selection to reduce dimensionality, handling class imbalance using SMOTE or ADASYN, hyperparameter optimization through Differential Evolution, and implementing machine learning and deep learning models. Metrics including Accuracy, Precision, Recall, and F1-score were used to evaluate performance, making sure that the assessment was strict and could be repeated.

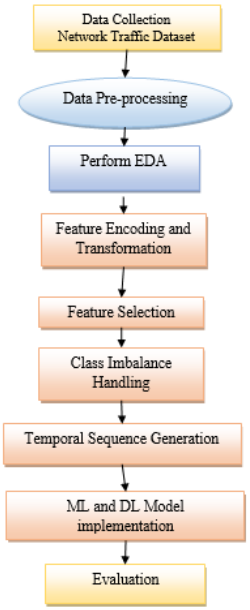

Figure 2 Proposed Research Flowchart

3.1 Data Collection

This study uses data from a publicly available network intrusion detection dataset that is provided on the Kaggle platform. We chose this dataset because it accurately represents real-world network traffic and is good for testing advanced intrusion detection methods. It has flow-based and packet-level features that record important network activity, such as packet counts, byte statistics, flow duration, port information, and traffic patterns over time. Each network flow instance is tagged with a categorical target variable that indicates either innocuous traffic or certain forms of attacks. This makes it possible to do multi-class intrusion detection analysis.
Link- https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset

The dataset was created from controlled network environments that mimic real-world communication patterns and a range of cyberattack situations. It has a lot of different types of intrusions, like denial-of-service assaults, probing operations, and other bad behaviors, as well as typical traffic. A key feature of the dataset is that it has a lot of dimensions and a big class imbalance. This is similar to operational networks where some sorts of attacks happen very rarely but are very dangerous to security. All data were sourced from open-source repositories without direct human intervention, assuring ethical adherence and reproducibility. The dataset was downloaded in a structured table format and kept safe for preprocessing, analysis, and testing.

3.2 Data Preprocessing

Before preprocessing and model building, a thorough data cleaning method was used to make sure that the network intrusion dataset was of high quality, reliable, and consistent. To get rid of duplicate data, duplicate records were found and deleted at first. This step was necessary to stop biased learning, since doing it over and over again can have a big effect on model training and make performance evaluation less accurate
When calculating features, several characteristics produced infinite values because of division-by-zero operations that are typical in network traffic monitoring. To avoid numerical instability or computational errors while the model was running, these infinite values were carefully found and replaced with missing values. After that, all of the missing values were replaced with zeros. This was a practical approach that kept the dataset's dimensionality while keeping the numbers stable and compatible with both machine learning and deep learning techniques
Also, column names were made consistent by getting rid of extra spaces at the beginning and end and making sure that all naming rules were the same. This stopped feature mismatches or execution problems from happening when building the pipeline, choosing features, and training the model. All of these data cleaning methods together made an organized, consistent, and error-free dataset that was a strong base for the next phases in transformation, optimization, and intrusion detection models.

3.3 Exploratory Data Analysis (EDA)

Before preprocessing and building the model, we did Exploratory Data Analysis (EDA) to get a better understanding of the network intrusion dataset's structural characteristics, statistical attributes, and class distribution. EDA helps find data that is not well distributed, is skewed, or has other problems that could affect how well a model works. EDA gives you important information that helps you choose the right data processing, scaling, and imbalance-handling strategies for constructing a strong intrusion detection framework by showing you class proportions, feature distributions, or variability patterns.

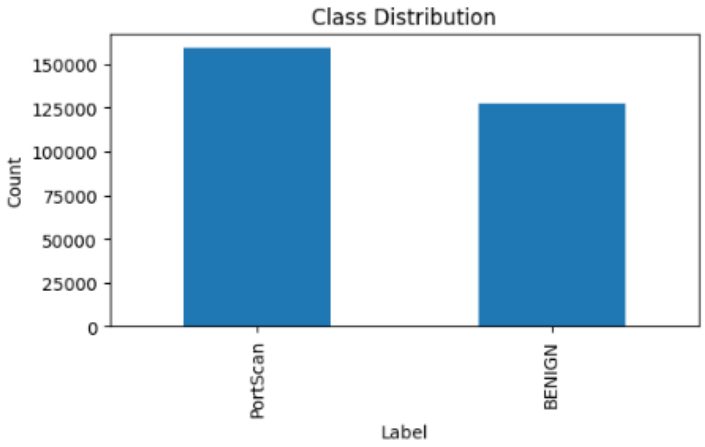

Figure 3 Class Distribution

This graphic shows how network traffic classifications are spread out using a bar chart. It clearly shows that there is a big class imbalance, with innocuous traffic and some forms of attacks taking up most of the data set, whereas minority attack classes are not well represented. This imbalance shows how important it is to use advanced resampling methods to find rare invasions more easily.

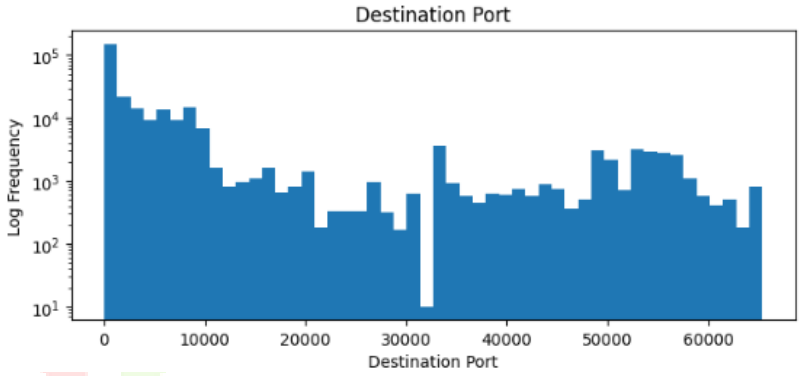

Figure 4 Destination port Graph

The destination port graph shows how often different destination ports are used in network traffic. The logarithmic scale shows a very skewed distribution, with only a few ports getting most of the traffic. This behavior shows how networks are really used in the real world and shows how important transformation techniques are for keeping feature variance stable.

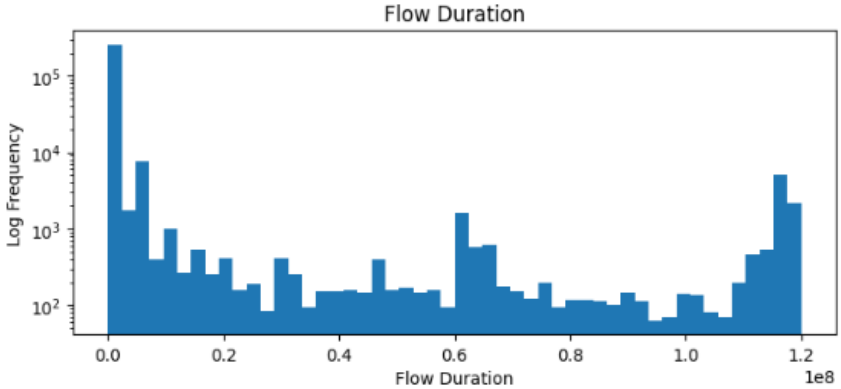

Figure 5 Flow Duration Graph

This figure shows how flow duration values are spread out over network traffic records. The distribution's strong tail shows that there are both short-lived and long-lasting connections. This kind of unpredictability sets innocuous traffic apart from attack behavior and makes it necessary to use power modifications to make the data less skewed.

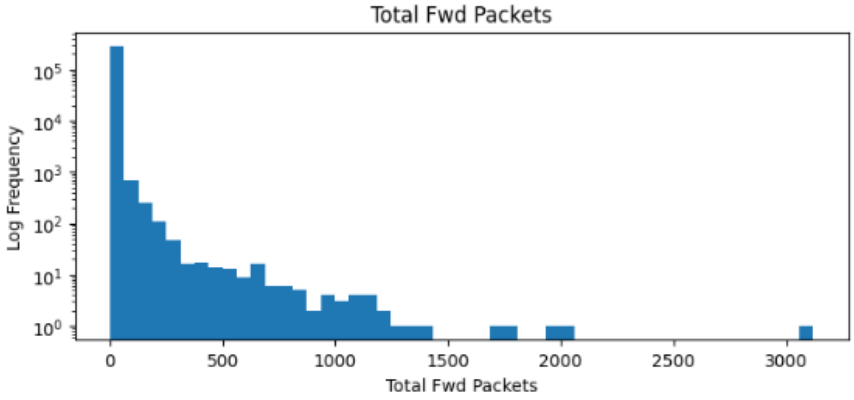

Figure 6 Total Fwd Packet graph

The total forward packets graph shows how packets are sent from one place to another. The plot is strongly right-skewed, which means that most flows have a modest number of packets and just a few flows have very big values. This trend shows how important it is to have strong scaling and feature normalization.

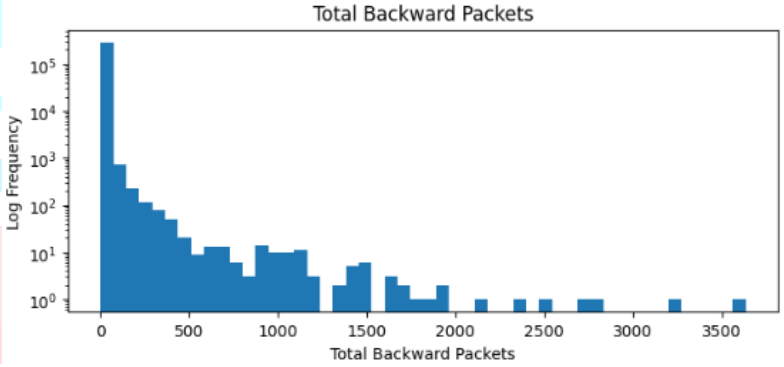

Figure 7 Total Backward packets Graph

This graph depicts how packets are transferred in the opposite direction of communication. The distribution is rather uneven, much like forward packets. This shows that network traffic has asymmetric communication behavior. This kind of asymmetry is especially helpful for recognizing the difference between typical network activity and attack patterns.

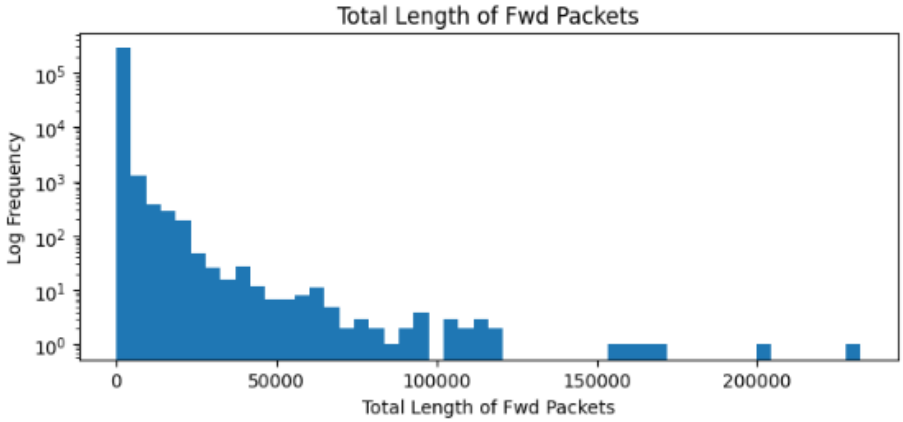

Figure 8 Total Length of Fwd Packets Graph

The graph shows how many bytes of data are in each flow of forward packets. The heavy-tailed distribution shows that there are bursts of high-volume traffic among mostly low-volume flows. These

differences are typical of several sorts of attacks, which shows how important it is to use logarithmic transformation and feature scaling.

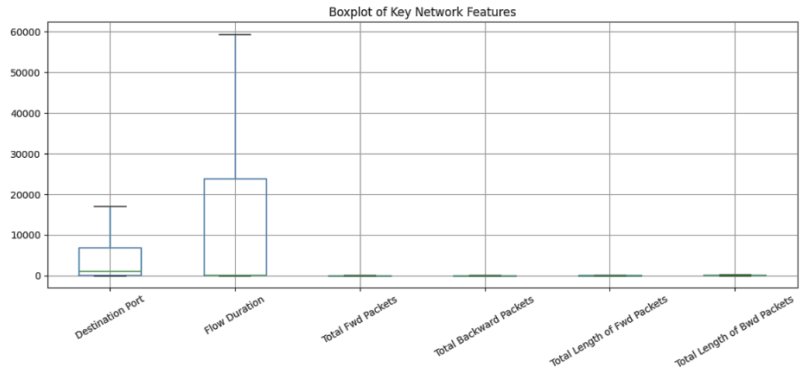

Figure 9 Boxplot of Key network Features

This boxplot shows the range and average of important network properties and points out any outliers. There are extreme values across many attributes, which shows that the traffic is acting strangely. The image backs up the idea of using strong preprocessing methods to lessen the effect of outliers during model training.

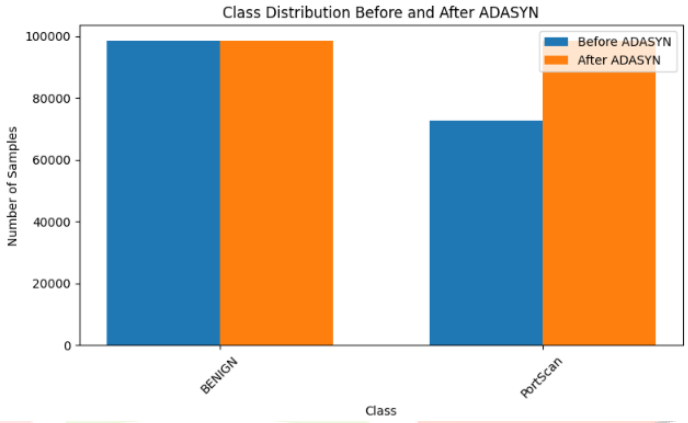

Figure 10 Class Distribution before and after ADASYN

3.4 Feature Encoding and Transformation

To create intrusion detection systems that work well, you need to be able to describe features well. This is especially true when dealing with network traffic data that is made up of both category and numerical variables. Raw network characteristics frequently display skewed distributions, scale inconsistencies, and category discrepancies that can adversely affect model learning and convergence. To solve these problems, a structured feature encoding and transformation method was used to make input representations more consistent and make the model easier to understand and more stable.

1. **Categorical Feature Encoding**

One-Hot Encoding was used to change the categorical attributes in the dataset. This method changes each category variable into a group of binary indicator variables. This lets machine learning models work with categorical data without forcing fake ordinal relationships. One-Hot Encoding changes a categorical feature X with k unique categories into a binary vector:

$$X_i = [x_{i1}, x_{i2}, \ldots . x_{ik}] \quad (1)$$

One-Hot Encoding keeps the nominal nature of network-related category features, including protocol kinds or traffic flags, and stops distance computations from being wrong when integer encoding is employed. This method increases model generalization by treating each category separately, which also allows for equitable contributions of categorical characteristics during training.

2. **Numerical Feature Transformation**

In network intrusion datasets, numerical features often include significant skewness, heavy-tailed distributions, and extreme variation because of unusual traffic patterns and bursts of attacks. The Yeo–Johnson power transformation was used on all numerical attributes to make these distributions more normal and keep the variance stable. The Yeo–Johnson transformation is better for real-world network traffic data because it can handle zero and negative values, which are typical in that kind of data. This transformation makes features more symmetrical, makes numbers more stable, and speeds up the process of both machine learning and deep learning models converging. The transformation makes sure that learning algorithms focus on significant patterns instead of noise by making distributional anomalies less common.

3. **Unified Preprocessing Pipeline**

A ColumnTransformer framework was used to construct a consistent preprocessing pipeline that made sure that the results were the same and could be repeated at all phases of the experiment. This pipeline combines the encoding of category features and the transformation of numerical features into one automated process. The pipeline gets rid of inconsistencies and stops transformation bias by using the same transformations on both the training and testing datasets. The pipeline's modular design also makes it easy to connect with resampling, feature selection, and optimization phases. This creates a strong and scalable preprocessing base for the intrusion detection framework.

3.5 Feature Selection Using Mutual Information

Datasets for network intrusions are usually high-dimensional, which means they have a lot of features that aren't very useful and make calculations harder and more likely to match too closely to the data. To solve this problem, we used a feature selection method based on Mutual Information (MI). Mutual Information measures how statistically dependent each characteristic is on the target variable. It does this without making any assumptions about how the data is distributed

We utilized the mutual_info_classif method to figure out how important each feature was to the intrusion class labels. These scores show how much each characteristic helps us guess the target variable. The SelectKBest method was then used to rank the features, and a fixed set of the top 300 highest-ranking features was kept for training the model. This selection criterion was selected to find a good middle ground between reducing the number of dimensions and keeping the information. To find the Mutual Information (MI) between a feature X and a class label Y:

$$MI(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \quad (2)$$

Feature selection was only done on the training dataset to keep information from leaking out. Then, the same subset of features was used on the testing data to make sure the evaluation was fair. This method cut down on the amount of computing power needed, made training more efficient, and increased generalization performance while keeping the most important network traffic features for accurate intrusion detection.

3.6 Handling Class Imbalance

Class imbalance is a common problem in intrusion detection datasets. There is a lot of innocuous traffic and some sorts of attacks, but not enough of the most important intrusion classes. If not dealt with, imbalance might cause learning algorithms to favor majority classes, which makes it hard to find rare but important assaults. To fix this problem, a multi-stage system for dealing with imbalances was put in place.

1. **Label Encoding**

Label encoding was used to turn the target class labels into numbers before employing resampling methods. This change makes sure that intrusion labels are categorical while still being able to work with resampling methods and classification models. Label encoding also makes it easier to do calculations quickly throughout the optimization and evaluation stages.

2. Synthetic Minority Over-sampling Technique (SMOTE)

To make synthetic samples for the minority classes, SMOTE was used on the training dataset. This method makes new instances by interpolating between existing minority samples and their closest neighbors. This adds to class representation without adding duplicate data points. SMOTE makes it less likely that a model will overfit and makes decision boundaries smoother than random oversampling. SMOTE makes classifiers better at finding modest intrusion patterns by increasing the density of the minority class.

Synthetic samples are generated as:

$$x_{new} = x_i + \delta \times (x_{nm} - x_i) \quad (3)$$

This creates realistic synthetic data points within minority class regions.

3. Adaptive Synthetic Sampling (ADASYN)

After SMOTE, Adaptive Synthetic Sampling (ADASYN) was used to make the class balance even better. ADASYN creates fake samples on the fly depending on the local density of occurrences in the minority class, with a concentration on examples that are hard to learn. This adaptive approach makes the classifier pay more attention to complex and overlapping areas of the feature space, which helps it tell the difference between attack and benign traffic. Using SMOTE and ADASYN together is a new way to handle imbalances that makes it easier to find rare types of intrusions. The quantity of synthetic samples produced for minority instance i is:

$$g_i = \frac{d_i}{\sum_{j=1}^{n} d_j} \times G \quad (4)$$

where:

- d_i is the local density of majority class neighbors,
- G is the total number of synthetic samples.

This ensures that harder-to-learn samples receive more emphasis.

3.7 Hyperparameter Optimization Using Differential Evolution

Choosing the right hyperparameters is very important for how well a classifier works, especially for kernel-based models like Support Vector Machines (SVM). Manual tuning and grid search approaches are costly in terms of computing and don't always do a good job of exploring the global parameter space. To get around these problems, we used Differential Evolution (DE), a population-based metaheuristic optimization approach, to find the best SVM hyperparameters. In Differential Evolution, new candidate solutions are created through mutation, which changes existing solutions by employing scaled vector differences, as described by:

$$V_i = x_{r1} + F \cdot (x_{r2} - x_{r3}) \quad (5)$$

DE was utilized to enhance the regularization parameter CCC and the kernel parameter γ connected with the Radial Basis Function (RBF) kernel. A stratified validation subset was created from the resampled training data to test possible solutions. The weighted F1-score was chosen as the fitness function to make sure that the performance was balanced between the majority and minority classes, which made the optimization goals match the needs of intrusion detection

The DE method used mutation, crossover, and selection operators over and over again to improve candidate solutions until they reached the best parameter settings. Mutation added variety by changing parameter vectors, crossover made it easier for solutions to share information, and selection made sure that the best candidates stayed alive. This evolutionary approach made it easy to explore across the search space and find the best hyperparameters for the whole system. The final SVM model was trained with the best-performing set of parameters once optimization had converged. This DE-optimized SVM is a major new feature of the proposed method. It combines evolutionary intelligence with strong

classification to improve accuracy, resilience, and generalization while the network is changing and not balanced.

3.8 Train–Test Split

After the features were encoded and changed, the dataset was split into training and testing sets using an 80–20 stratified split approach. Stratification made sure that the percentage of each traffic class stayed the same in both subsets. This is especially significant because network intrusion data has a very uneven class distribution. Keeping the original class distribution makes it possible to fairly and accurately measure how well the model generalizes. After that, all activities, such as addressing class imbalance, selecting features, optimizing hyperparameters, and training the model, were exclusively done on the training dataset. During these steps, the testing dataset was kept entirely hidden to stop information from leaking and skewed performance estimates. This rigorous way of dividing up data makes experimental results more reliable and makes sure that the performance measures given accurately show how well the suggested intrusion detection framework will work in the actual world.

3.9 ML Models Implementation

The following machine learning classifiers were implemented and evaluated:

1. **Support Vector Machine (SVM):** We used Differential Evolution to find the best hyperparameters for the Support Vector Machine classifier, which improved its ability to find things in changing network settings. The RBF kernel allowed for learning nonlinear decision boundaries, which made the model good at recognizing complicated incursion patterns. DE-based optimization made sure that the parameter space was explored all around the world, which made it more resilient and better at generalizing. The SVM worked well as a baseline and was able to tell the difference between attacks from both the majority and minority classes.
2. **k-Nearest Neighbors (KNN):** The k-Nearest Neighbors model was set up to use distance-based weighting to give more weight to samples that are closer to the one being classified. This method let the classifier learn how to operate with local data structures and pick up on small changes in how network traffic behaves. KNN was good for modeling complex distributions because it didn't use parameters, but it only worked well when feature scaling or dimensionality reduction were done correctly.
3. **Multi-Layer Perceptron (MLP):** The Multi-Layer Perceptron was made with fully connected hidden layers so that it could learn nonlinear feature representations from network traffic data. Activation functions and regularization methods made it possible to learn patterns well while preventing overfitting. MLP showed that it could easily adapt to high-dimensional information and find complex connections between network parameters in the intrusion detection challenge.
4. **Extra Trees Classifier:** The Extra Trees Classifier used a group of random decision trees to make classification more stable and less variable. The model improved generalization and reduced overfitting by adding randomness to feature selection and split thresholds. This ensemble method worked well for dealing with large intrusion datasets and capturing a wide range of assault patterns.
5. **XGBoost:** We choose XGBoost because it has a strong gradient-boosting framework that builds decision trees one at a time to fix mistakes made in the past. It has good prediction performance because to its regularization algorithms and ability to handle vast amounts of data quickly. XGBoost was able to accurately simulate complicated interactions between features and was quite accurate in situations when it had to detect multiple types of intrusions.
6. **CatBoost:** CatBoost was used since it is good at handling categorical features and doesn't overfit. Its ordered boosting method reduced target leakage and made generalization better. CatBoost worked well on multi-class network traffic data and made reliable, high-quality predictions without needing a lot of feature preprocessing.

All models were trained on balanced training data and evaluated using identical metrics to ensure fair comparison.

3.10 Temporal Sequence Generation

We used a sliding window approach to construct temporal sequences that would help with sequential and temporal modeling of network traffic behavior. The create_sequences method turned the cleaned and changed tabular dataset into fixed-length sequences. Each sequence had network flow records that were in chronological order. This method helped the model find time-based dependencies, traffic evolution patterns, or short-term correlations that are important for finding complicated and stealthy infiltration behaviors. The model learned how to forecast how the network would behave in the future by assigning the right target label to each generated sequence based on the class of the next time step. To create overlapping sequences, the sliding window was shifted over the dataset in small steps. This added more training examples and made learning more robust. This temporal representation improves the performance of sequence-based learning models by adding contextual and dynamic information to static feature analysis.

3.11 Deep Learning Models Implementation

We used deep learning models to automatically discover complicated, non-linear patterns from high-dimensional or sequential network traffic data. By using spatial feature extraction or temporal dependency modeling, these models improve intrusion detection, flexibility, and resistance to changing cyberattacks in networks that are always changing.

1. **Convolutional Neural Network (CNN):** A one-dimensional Convolutional Neural Network (1D-CNN) was used to automatically learn how to tell the difference between different types of network traffic data. The model used convolutional layers to find local spatial patterns and correlations between nearby features, which are important for telling the difference between normal and malicious traffic flow. Multiple convolution filters made it possible to extract features at different scales, which made it easier to detect different types of attacks. Global average pooling was added to keep important information while making the model less complex and less likely to overfit. This design made learning easier by using fewer parameters, improving generalization, and speeding up convergence. This made the CNN a good choice for high-dimensional intrusion detection data.
2. **Long Short-Term Memory (LSTM):** We used the Long Short-Term Memory (LSTM) network to find long-term temporal dependencies in sequential network traffic flows. Its gated architecture effectively solved the vanishing gradient problem, which made it possible to learn from long-term time contexts. LSTM layers worked with time-ordered sequences made by sliding windows, which helped the model see changing attack patterns and delayed intrusion behaviors. Batch normalization was used to make training more stable, while dropout regularization was used to stop overfitting by turning off neurons at random times during training. This combination made the LSTM model more robust, allowing it to generalize well across a wide range of intrusion scenarios that change over time.
3. **Hybrid CNN–GRU / TCN-Based Models:** Hybrid deep learning architectures that combine convolutional layers with recurrent units were made to capture both the spatial and temporal features of network traffic at the same time. Convolutional layers were good at finding important patterns and correlations between features, but Gated Recurrent Units (GRU) or Temporal Convolutional Networks (TCN) approximated temporal connections with less computing complexity. Compared to LSTMs, GRUs trained faster and had fewer parameters. TCNs, on the other hand, allowed for parallel processing and long receptive fields. By combining short-term feature patterns with long-term temporal behavior, this hybrid architecture improved detection accuracy. This made the models quite good at finding intrusions in real time.

Table 2 Hyperparameter Configuration of Deep Learning Models

Parameter	1D CNN Model	LSTM Model	CNN–GRU / TCN-Based Model
Batch Normalization	Yes	Yes	No
Dropout Rate	0.4	0.3	0.3
Output Layer	Dense (Softmax)	Dense (Softmax)	Dense (Softmax)
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.0005
Gradient Clipping	No	No	ClipNorm = 1.0
Loss Function	Sparse Categorical Cross-Entropy	Sparse Categorical Cross-Entropy	Sparse Categorical Cross-Entropy
Batch Size	128	128	128
Epochs	100	100	100
Class Weights	Yes	Yes	No
Evaluation Metrics	Accuracy, Precision, Recall, F1-score	Accuracy, Precision, Recall, F1-score	Accuracy, Precision, Recall, F1-score

3.12 Performance Evaluation Metrics

We used Accuracy, Precision, Recall, and F1-score measures to carefully check how well all of the machine learning and deep learning models worked. To make sure that all network traffic categories were evaluated fairly, macro-averaging was used to treat each class equally. This helped reduce bias from class distributions that weren't balanced. Confusion matrices were created to analyze performance on a class-by-class basis in depth, showing both accurate and incorrect classifications. Using the argmax function, predicted probability distributions were turned into discrete class labels for deep learning models. This made it possible to compare them with true labels. This thorough evaluation framework showed how well the model worked overall, how strong it was, and how well it worked for real-time intrusion detection.

4. Results and Discussion

This section provides a thorough examination of the experimental results derived from the assessment of machine learning and deep learning models used to the network intrusion detection dataset. The findings include metrics for how well the model works, information about which features are most important, and the effects of preprocessing, feature selection, and handling class imbalance. Comparative discussions emphasize the advantages, drawbacks, and practical ramifications of each model in identifying various attack patterns. The study also puts the results in context by looking at the properties of the dataset. It focuses on how robust, generalizable, and successful the model is in real-world network situations.

1) Accuracy

To find out how accurate a model is, you can find the percentage of correctly identified instances out of the total samples. High accuracy in intrusion detection means that the model can tell the difference between good and bad traffic in all types of attacks.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

2) Precision

Precision measures the percentage of true positive predictions out of all samples that were projected to be positive. In IDS, high accuracy means that the model reduces false alarms, which means that attacks that the system flags are very likely to be real threats. This cuts down on unnecessary warnings for network admins.

$$Precision = \frac{TP}{TP+FP} \tag{7}$$

3) Recall

Recall, or sensitivity, is the percentage of true positive cases that the model properly identifies. High recall is very important for intrusion detection since it makes sure that most attacks are found, even ones that happen seldom or are hard to find. This makes the system more secure.

$$Recall = \frac{TP}{TP+FN} \tag{8}$$

4) F1 Score

The F1-Score is the harmonies of precision and recall, which means it balances false positives or false negatives. A high F1-score in IDS evaluation means that the model can find threats accurately and with few false alarms, making it strong enough for real-world cybersecurity use.

$$F1-score = \frac{2}{\frac{1}{precision}+\frac{1}{recall}} \tag{9}$$

We tested the machine learning models used in this study to see how well they worked at finding network intrusions. We used macro-averaging to make sure that all classes were treated fairly while calculating the key metrics Accuracy, Precision, Recall, and F1-score. This evaluation shows how well each model can find both frequent and unique sorts of attacks.

Table 3 Machine Learning Models Performance Evaluation

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-score (Macro)
MLP	0.9998	0.9998	0.9998	0.9998
KNN	0.9999	0.9999	0.9999	0.9999
Extra Trees	0.9999	0.9999	0.9999	0.9999
XGBoost	0.9999	0.9999	0.9999	0.9999
CatBoost	0.9999	0.9999	0.9998	0.9999

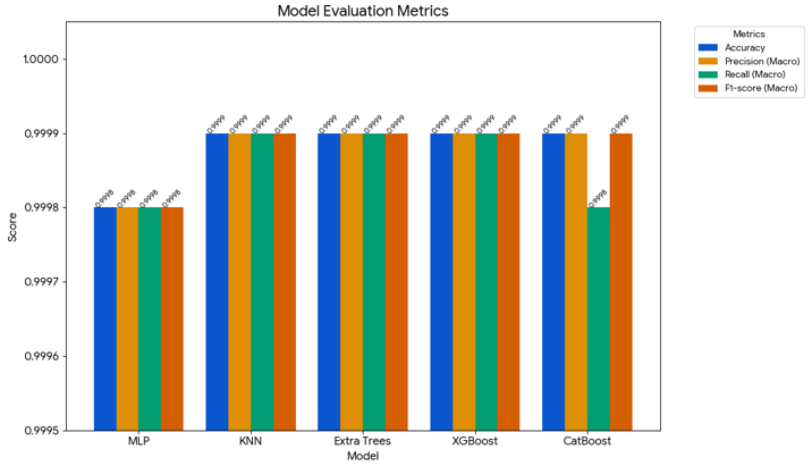

Figure 11 ML Models Performance Evaluation Graph

Table 3 shows how well the MLP, KNN, Extra Trees, XGBoost, and CatBoost classifiers work. All of the models were very accurate, with KNN, Extra Trees, and XGBoost getting 0.9999, which means they almost perfectly found network traffic classes. The Precision and Recall numbers are also high, which shows that the models can properly identify both the majority and minority classes. The F1-score, which balances precision and recall, shows that these classifiers are very strong, with only small differences between models. CatBoost has a little lower F1-score (0.9999 vs. 0.9998) because it handles imbalanced attack classes a little differently. Figure shows these indicators in a way that makes it easy to compare the performance of different classifiers and shows that ensemble and distance-based models are better at accurately identifying complicated network data.

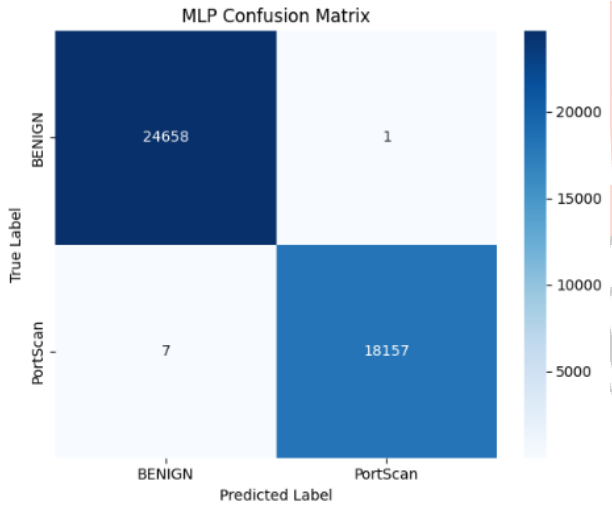

Figure 12 MLP Confusion Matrix

This matrix shows how well the Multi-Layer Perceptron (MLP) model predicts classes. The values on the diagonal show correctly classified instances, and the values off the diagonal show incorrectly classified instances. The almost perfect diagonal dominance shows that MLP is quite accurate and works well with all types of network traffic, both good and bad.

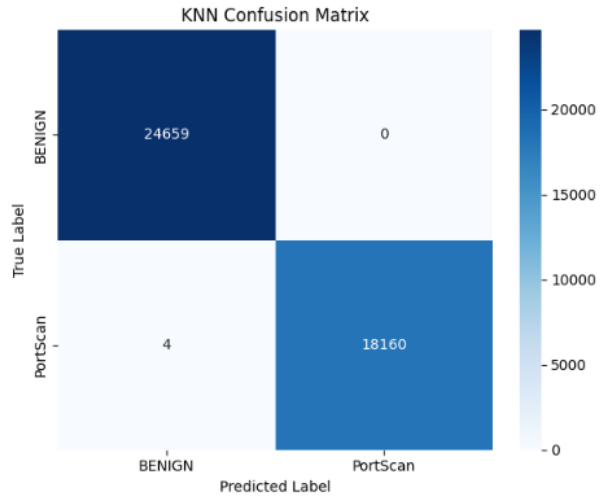

Figure 13 KNN Model Confusion Matrix

The K-Nearest Neighbors (KNN) confusion matrix reveals that the predictions are very consistent, with most of the examples being properly classified along the diagonal. Few off-diagonal entries mean that misclassifications don't happen very often. This shows that KNN is good at finding local data structures and telling the difference between typical and different forms of intrusions in the dataset.

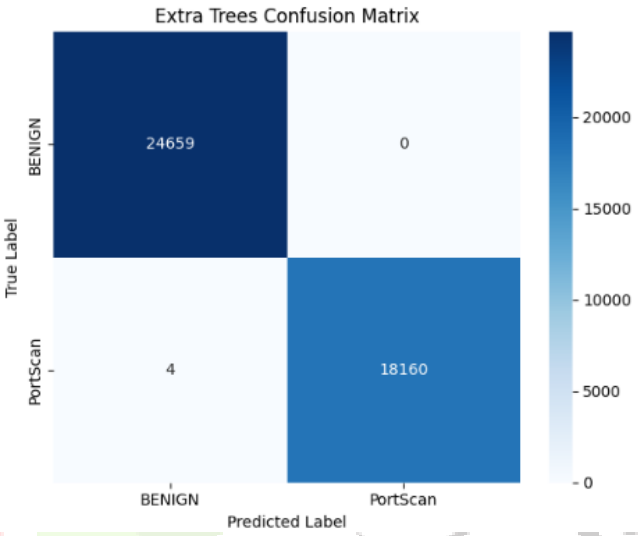

Figure 14 Confusion Matrix of Extra trees Model

This confusion matrix for the Extra Trees model shows how strong the ensemble is at correctly identifying network traffic. Maximizing the diagonal elements shows that all classes are correctly identified. The small number of misclassifications off the diagonal shows that the model generalizes well and has less variation because the trees were built randomly.

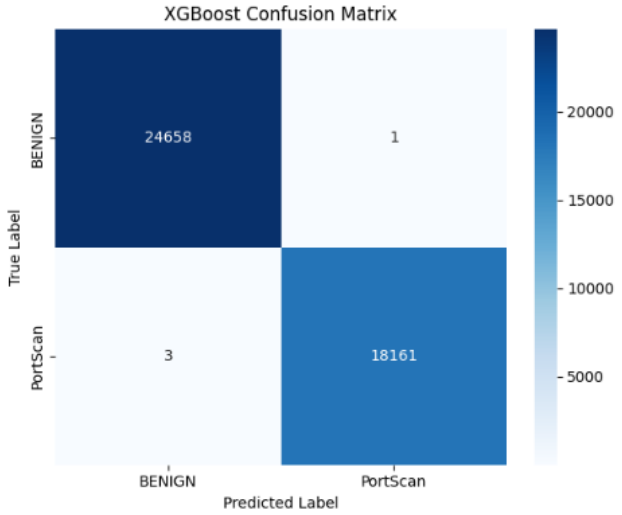

Figure 15 confusion Matrix of XGBoost model

The XGBoost confusion matrix shows that almost all cases are successfully mapped to their true labels, which means that the class predictions are very accurate. The little off-diagonal counts show that the model learned how to handle complicated feature interactions and multi-class imbalances in network incursion data well.

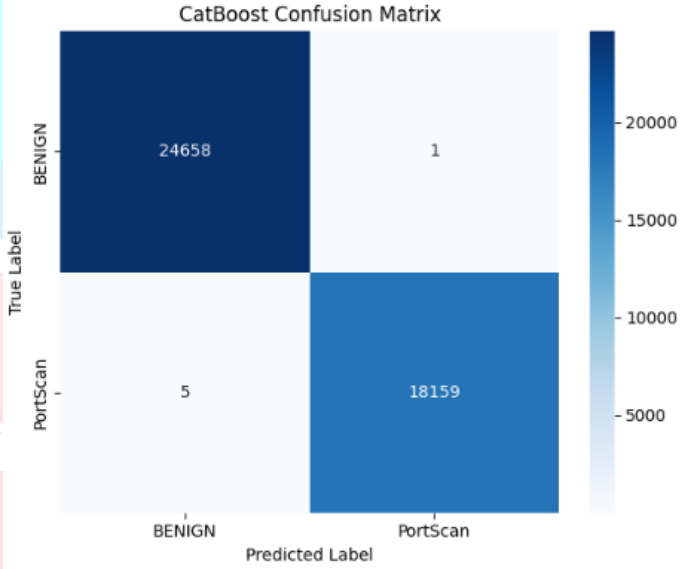

Figure 16 Confusion Matrix of CatBoost Model

The confusion matrix for CatBoost reveals that it does a great job of classifying things because the diagonal values are so high. The few examples that were misclassified show that there are some little problems with overlapping class borders. The matrix shows that CatBoost is good at processing categorical information and making steady, accurate predictions for all types of network traffic.

Table 4 Deep Learning Models Performance Evaluation

Model	Accuracy	Precision	Recall	F1-score
TCN–GRU	0.6065	0.6274	0.6065	0.5897
LSTM	0.6222	0.6960	0.6222	0.5829
1D CNN	0.9998	0.9999	0.9998	0.9998

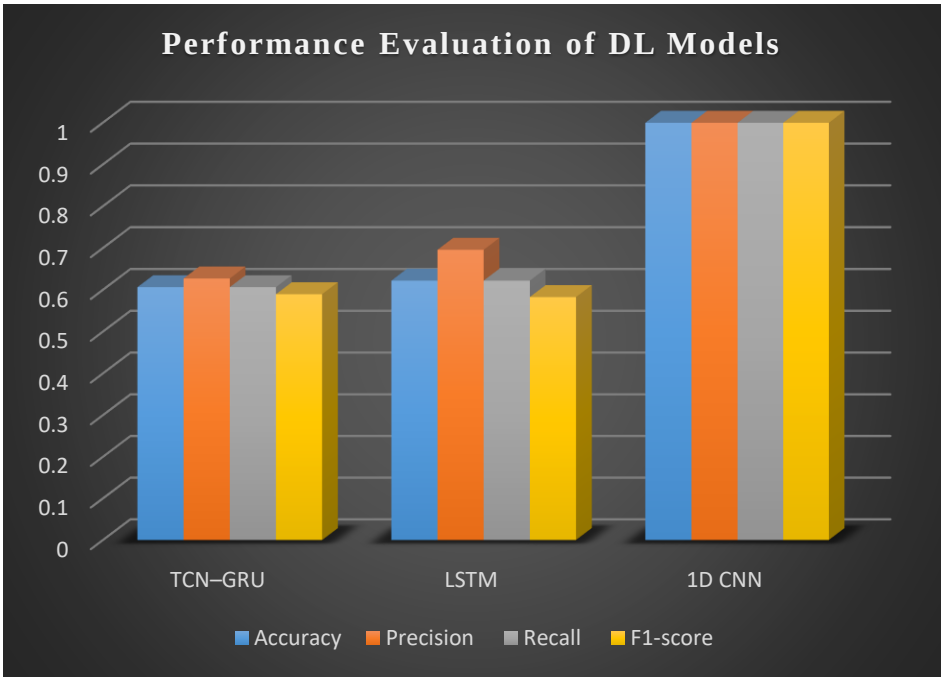

Figure 17 Deep Learning Models Performance Evaluation Graphs

Table 4 displays the results of testing the TCN–GRU, LSTM, and 1D CNN deep learning models on the network intrusion detection dataset. We used the Accuracy, Precision, Recall, and F1-score measures to fully test how well the model could tell the difference between regular and malicious data. The TCN–GRU model got 60.65% correct, 62.74% correct, and 58.97% F1-score, which means it could identify some classes properly but not all of them. The LSTM model did somewhat better, with an accuracy of 62.22% and a precision of 69.60%. This shows that it was better at capturing temporal dependencies, but the F1-score stayed the same (58.29%), which means that it still had trouble dealing with class imbalance

On the other hand, the 1D CNN did far better than both recurrent-based models, getting almost perfect accuracy (99.98%) and F1-score (99.98%). This shows that it is very good at finding important spatial patterns in high-dimensional information. This suggests that convolutional architectures successfully identified important patterns for intrusion detection in tabular network traffic data, outperforming temporal sequence modeling methods. Figure shows these data in a way that makes it clear that the 1D CNN model is better than the other models in terms of performance.

5. CONCLUSION

This study put out a thorough and flexible intrusion detection framework that combines swarm intelligence with cutting-edge machine learning and deep learning methods to solve the problems of network security that are always changing and happening in real time. A strong preprocessing pipeline that used categorical encoding, numerical transformation, or mutual information–based feature selection made sure that feature representations were stable and useful while cutting down on dimensionality and computing overhead. The combined use of SMOTE and ADASYN greatly reduced severe class imbalance, making it much easier to find rare or high-impact attack classes. Differential Evolution–based hyperparameter optimization was very important for making classifiers more robust since it let the parameter space be searched globally, which was especially useful for kernel-based models. The experimental evaluation showed that traditional machine learning models like KNN, Extra Trees, XGBoost, and CatBoost worked almost perfectly, which shows that they can generalize well to balanced network traffic data. The 1D Convolutional Neural Network was the best deep learning model, with an accuracy of 99.98%, a precision of 99.99%, a recall of 99.98%, and an F1-score of 99.98%. These results reveal that the CNN is better at finding discriminative spatial patterns in high-dimensional network features than recurrent and hybrid temporal models like LSTM and TCN–GRU, which did not perform as well. The results show that combining swarm intelligence-driven optimization with carefully planned learning architectures provides an intrusion detection system that is very

accurate, strong, and able to grow. The suggested system is great for real-time use and is a good base for future additions like adaptive thresholds, online learning, and changing cyberattack circumstances.

REFERENCES

[1] A. Karim and S. Ali, "INTELLIGENT INTRUSION DETECTION AND DATA PROTECTION IN INFORMATION SECURITY USING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING TECHNIQUES," vol. 3138, pp. 818–828, 2025.

[2] Oluwafemi Oloruntoba, "AI-Driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise it environments," *World J. Adv. Res. Rev.*, vol. 25, no. 2, pp. 1558–1580, 2025, doi: 10.30574/wjarr.2025.25.2.0534.

[3] O. S. Ndibe, "AI-Driven Forensic Systems for Real-Time Anomaly Detection and Threat Mitigation in Cybersecurity Infrastructures," *Int. J. Res. Publ. Rev.*, vol. 6, no. 5, pp. 389–411, 2025, doi: 10.55248/gengpi.6.0525.1991.

[4] W. El-Shafai, A. T. Azar, and S. Ahmed, "AI-Driven Ensemble Classifier for Jamming Attack Detection in VANETs to Enhance Security in Smart Cities," *IEEE Access*, vol. 13, no. February, pp. 50687–50713, 2025, doi: 10.1109/ACCESS.2025.3552544.

[5] K. C. Chaganti, "A Scalable, Lightweight AI-Driven Security Framework for IoT Ecosystems: Optimization and Game Theory Approaches," *IEEE Access*, vol. 13, no. May, pp. 72235–72247, 2025, doi: 10.1109/ACCESS.2025.3558623.

[6] A. A. Hammad and F. T. Jasim, "Adaptive Cyber Defense using Advanced Deep Reinforcement Learning Algorithms: A Real-Time Comparative Analysis," *J. Comput. Theor. Appl.*, vol. 2, no. 4, pp. 523–535, 2025, doi: 10.62411/jcta.12560.

[7] K. Tallam, "CyberSentinel: An Emergent Threat Detection System for AI Security," pp. 1–27, 2025, [Online]. Available: http://arxiv.org/abs/2502.14966

[8] T. Ali, "Next-generation intrusion detection systems with LLMs : real-time anomaly detection, explainable AI, and adaptive data generation," *Laturi.Oulu.Fi*, no. June, 2024, [Online]. Available: https://oulurepo.oulu.fi/handle/10024/51067

[9] Olakunle Abayomi Ajala, Chinwe Chinazo Okoye, Onyeka Chrisanctus Ofodile, Chuka Anthony Arinze, and Obinna Donald Daraojimba, "Review of AI and machine learning applications to predict and Thwart cyber-attacks in real-time," *Magna Sci. Adv. Res. Rev.*, vol. 10, no. 1, pp. 312–320, 2024, doi: 10.30574/msarr.2024.10.1.0037.

[10] M. Bai and X. Fang, "Machine Learning-Based Threat Intelligence for Proactive Network Security," *Integr. J. Sci. Technol.*, vol. 1, no. 2, pp. 1–10, 2024, [Online]. Available: https://creativecommons.org/licenses/by/4.0/deed.en

[11] A. Alhammadi *et al.*, "Artificial Intelligence in 6G Wireless Networks: Opportunities, Applications, and Challenges," *Int. J. Intell. Syst.*, vol. 2024, 2024, doi: 10.1155/2024/8845070.

[12] M. Farooq and M. Hassan Khan, "AI-Driven Network Security: Innovations in Dynamic Threat Adaptation and Time Series Analysis for Proactive Cyber Defense," *Int. J. Wirel. Microw. Technol.*, vol. 14, no. 2, pp. 17–26, 2024, doi: 10.5815/ijwmt.2024.02.02.

[13] S. Atakishiyev, M. Salameh, H. Yao, and R. Goebel, "Explainable Artificial Intelligence for Autonomous Driving: A Comprehensive Overview and Field Guide for Future Research Directions," *IEEE Access*, vol. 12, no. July, pp. 101603–101625, 2024, doi: 10.1109/ACCESS.2024.3431437.

[14] M. Aminu and A. Akinsanya, "Enhancing Cyber Threat Detection through Real-time Threat Intelligence and Adaptive Defense Mechanisms," *Int. J. Comput. Appl. Technol. Res.*, no. July, pp. 10–27, 2024, doi: 10.7753/ijcatr1308.1002.

[15] V. Ramamoorthi and W. Kluwer, "Anomaly Detection and Automated Mitigation for Microservices Security with AI," no. June, p. 222, 2024, [Online]. Available: https://www.researchgate.net/publication/386567677

[16] M. M. Khan, "Developing AI-Powered Intrusion Detection System for Cloud Infrastructure," *J. Artif. Intell. Mach. Learn. Data Sci.*, vol. 2, no. 1, pp. 1074–1080, 2024, doi: 10.51219/jaimld/mohammed-mustafa-khan/255.

[17] S. Racherla, P. Sripathi, N. Faruqui, M. Alamgir Kabir, M. Whaiduzzaman, and S. Aziz Shah, "Deep-IDS: A Real-Time Intrusion Detector for IoT Nodes Using Deep Learning," *IEEE Access*, vol. 12, no. April, pp. 63584–63597, 2024, doi: 10.1109/ACCESS.2024.3396461.

[18] O. Ebong, A. Edet, A. Uwah, and N. Udoetor, "Comprehensive Impact Assessment of Intrusion Detection and Mitigation Strategies Using Support Vector Machine Classification," *Res. J. Pure Sci. Technol. E*, vol. 7, no. 2, pp. 50–69, 2024, doi: 10.56201/rjpst.v7.no2.2024.pg50.69.

[19] C. Hernández, M. López, J. García, and C. Vander Peterson, "Optimizing collaborative intelligence systems for end-to-end cybersecurity monitoring in global supply chain networks," no. March 2024, 2024, [Online]. Available: https://www.researchgate.net/profile/Carrie-Peterson-6/publication/387948666_Optimizing_Collaborative_Intelligence_Systems_for_End-to-End_Cybersecurity_Monitoring_in_Global_Supply_Chain_Networks/links/6783c3cc8210a977a184bfb3/Optimizing-Collaborative-Inte

[20] J. Kipongo, T. G. Swart, and E. Esenogho, "Artificial Intelligence-Based Intrusion Detection and Prevention in Edge-Assisted SDWSN With Modified Honeycomb Structure," *IEEE Access*, vol. 12, no. January, pp. 3140–3175, 2024, doi: 10.1109/ACCESS.2023.3347778.

[21] R. Ranpara, O. Alsalman, O. P. Kumar, and S. K. Patel, "A simulation-driven computational framework for adaptive energy-efficient optimization in machine learning-based intrusion detection systems," *Sci. Rep.*, vol. 15, no. 1, pp. 1–13, 2025, doi: 10.1038/s41598-025-93254-4.

[22] P. Moriano, S. C. Hespeler, M. Li, and M. Mahbub, "Adaptive anomaly detection for identifying attacks in cyber-physical systems: A systematic literature review," *Artif. Intell. Rev.*, vol. 58, no. 9, 2025, doi: 10.1007/s10462-025-11292-w.

[23] N. Mohamed, "Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms," *Knowl. Inf. Syst.*, vol. 67, no. 8, pp. 6969–7055, 2025, doi: 10.1007/s10115-025-02429-y.

[24] S. I. Popoola, Y. Tsado, A. A. Ogunjinmi, E. Sanchez-Velazquez, Y. Peng, and D. B. Rawat, "Multi-Stage Deep Learning for Intrusion Detection in Industrial Internet of Things," *IEEE Access*, vol. 13, no. April, pp. 60532–60555, 2025, doi: 10.1109/ACCESS.2025.3557959.

[25] Adewale Daniel Sontan and Segun Victor Samuel, "The intersection of Artificial Intelligence and cybersecurity: Challenges and opportunities," *World J. Adv. Res. Rev.*, vol. 21, no. 2, pp. 1720–1736, 2024, doi: 10.30574/wjarr.2024.21.2.0607.

[26] L. Qudus, "Leveraging Artificial Intelligence to Enhance Process Control and Improve Efficiency in Manufacturing Industries," *Int. J. Comput. Appl. Technol. Res.*, no. March, pp. 17–38, 2025, doi: 10.7753/ijcatr1402.1002.

[27] A. Berguiga, A. Harchay, and A. Massaoudi, "HIDS-IoMT: A Deep Learning-Based Intelligent Intrusion Detection System for the Internet of Medical Things," *IEEE Access*, vol. 13, no. February, pp. 32863–32882, 2025, doi: 10.1109/ACCESS.2025.3543127.

[28] V. Z. Mohale and I. C. Obagbuwa, "A systematic review on the integration of explainable artificial intelligence in intrusion detection systems to enhancing transparency and interpretability in cybersecurity," *Front. Artif. Intell.*, vol. 8, no. January, pp. 1–10, 2025, doi: 10.3389/frai.2025.1526221.

[29] N. Srinivasan, "Artificial Intelligence in IoT Security: Review of Advancements, Challenges, and Future Directions," *Int. J. Innov. Technol. Explor. Eng.*, vol. 13, no. 7, pp. 14–20, 2024, doi: 10.35940/ijitee.g9911.13070624.

[30] Y. A. Farrukh, S. Wali, I. Khan, and N. D. Bastian, "AIS-NIDS: An intelligent and self-sustaining network intrusion detection system," *Comput. Secur.*, vol. 144, pp. 1–16, 2024, doi: 10.1016/j.cose.2024.103982.