# Development and Implementation of Machine Learning-Based Anomaly Detection System

Lee Weng Hong

Sunway University, Selangor 47500, Malaysia

**Abstract.** Intrusion Detection Systems (IDS) are critical for safeguarding network infrastructures against various cyberattacks. While many researchers have shown that Machine Learning (ML)-based IDS can identify malicious traffic in synthetic datasets, few have evaluated their performance against real-world network traffic. This paper explores the impact of preprocessing techniques on model performance and evaluates how well an IDS trained on a synthetic dataset performs in real-world settings. We propose that incorporating a Local Baseline Profile (LBP) during training improves model performance. Our experiments include dataset size reduction, SMOTE oversampling, attack class grouping, dataset scaling, and feature reduction. We created multiple IDS models using the best preprocessing combinations and tested them against locally captured network traffic, containing both benign and attack traffic. Our findings show that adding LBP during training significantly improved detection rates for Decision Tree and Random Forest models in Brute Force attacks. However, our proposed IDS has limitations in detecting a wider range of network attacks, especially more complex and unseen ones. This highlights the need for enhanced training data, advanced feature extraction techniques, and adaptive learning models to improve IDS performance.

**Keywords:** Intrusion Detection Systems, Machine Learning, Preprocessing, Real-world Cyber Security Threats, IDS Performance.

## 1 Introduction

The development of information technology in recent years has enabled fast communication, facilitating information exchange across distances. One key area is the Internet of Things (IoT), which involves devices with sensors that collect and exchange data over the internet. However, this growth in connectivity brings security challenges, with increasing IoT attacks. Problematically, IoT devices are particularly vulnerable due to their interconnected nature [1]. In 2022, the number of IoT cyber attacks worldwide totaled 112 million, 80 million more cases than in the year 2018 [2]. Attackers use compromised IoT devices for various attacks, compromising the confidentiality, integrity, and availability of networking devices. As a result, this necessitates the prevention or detection of such attacks using various techniques.

An intrusion detection system (IDS) is a software or hardware system that monitors network traffic or host activities for signs of intrusions [3]. Unlike traditional firewalls, IDS can dynamically filter out unwanted activities, ensuring device availability, integrity, and confidentiality [3]. IDS can be Network-based (NIDS), analyzing traffic at strategic points, or Host-based (HIDS), monitoring system resources for anomalies. Additionally, IDS can be Signature-based (SIDS), detecting known attack patterns, or Anomaly-based (AIDS), identifying deviations from normal behavior [4, 5]. AIDS uses ML models to establish a baseline of normal network behavior, subsequently identifying deviations from this baseline [5]. This makes AIDS excel at detecting previously unknown attacks, but creating a normal baseline profile that has low false positive rates can be difficult [2, 3, 4].

Machine learning (ML) is a subset of artificial intelligence (AI) that enables machines to learn from data and aid in the decision-making process. The integration of ML in IDS helps detect new, unknown attacks by learning from data and identifying patterns [2]. ML-based IDS, such as AIDS can adapt to new threats by continuously updating its database.

The performance of an ML-based IDS depends on the quality and relevance of its training data. Researchers have explored preprocessing techniques like hyperparameter tuning, SMOTE oversampling, feature selection, and normalization to improve the quality of training data and the overall detection performance. One of the widely used datasets is CICIDS2017 but it suffers from class imbalance [6, 7]. Many researchers address this by using SMOTE oversampling to generate new samples of minority classes [6, 8].

Researchers often test ML models using synthetic datasets, which may not reflect real-world scenarios. For example, newer devices generating different traffic patterns compared to the training data can create a different baseline profile. This could lead to increased false positives and false negatives due to deviations from the expected traffic pattern. In summary, rapid changes in network conditions, including cybersecurity threats, may render models trained on existing datasets inadequate for real-world scenarios, potentially yielding less meaningful results. Therefore, this study compares the performance of various preprocessing techniques and analyzes their effectiveness against real-world traffic, aiming to understand how different ML models respond to variations between trained and real networks generated baseline profiles.

## 2 Methodology

This section outlines the methodology for designing and evaluating ML-based IDS for network security. We present comprehensive details of our performance assessment setup, including the dataset employed for training, the training methodology, descriptions of the MS models utilized along with preprocessing techniques, and the methodology for capturing and analyzing network traffic within a local area network (LAN). Lastly, we elaborate on the metrics employed to evaluate the effectiveness of our Intrusion Detection System (IDS). Fig 1 shows the proposed IDS framework.

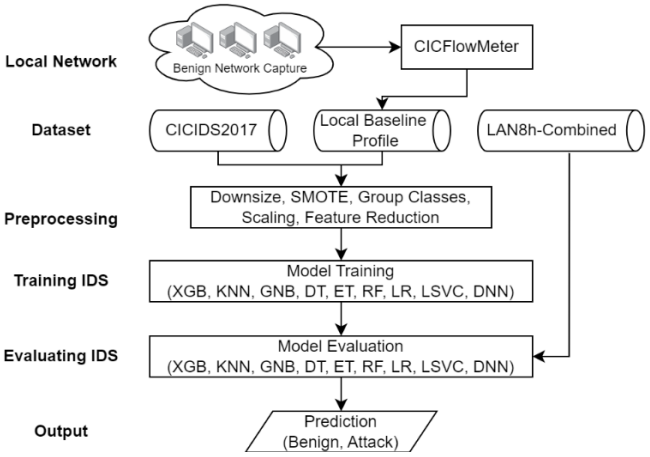


**Fig. 1.** Proposed IDS framework.

### 2.1 Performance Assessment

Performance assessment was done on a 64-bit Windows 10 machine with an AMD Ryzen 5-3600 six-core CPU (12 threads) at 4.40 GHz and 32GB of memory. Each model's performance was evaluated using Accuracy, Precision, Recall, F1-score, and Confusion Matrix.

### 2.2 Datasets and Machine Learning Models

The CICIDS2017 dataset, created by the Canadian Institute for Cybersecurity, was used. It contains 2,830,743 records with 80 features, representing various types of network attacks and normal traffic [9]. This dataset was chosen for its realistic and diverse network traffic data, reflecting current network security trends and challenges.

In total, nine ML models were used. These models were implemented in Python (version 3.10.10) using the scikit-learn library (version 1.3.1), while hyperparameter tuning was performed using the Optuna library (version 3.4.0). Table 1 lists the models and the hyperparameters utilized for this study.

**Table 1.** Selection of Machine Learning Models and Hyperparameters.

| Model | Hyperparameters |
|---|---|
| Logistic Regression (LR) | C = 0.36585696635446396, max_iter = 868, solver = "lbfgs" |
| K-Nearest Neighbors (KNN) | n_neighbors = 21 |
| Gaussian Naïve Bayes (GNB) | var_smoothing = 9.437310900762216e-08 |
| Linear Support Vector Classification (LSVC) | C = 1.0000346600564648e-05, max_iter = 709 |
| Decision Trees (DT) | max_depth = 10, min_samples_split = 13, min_samples_leaf = 3 |

| | |
|---|---|
| XGBoost (XGB) | n_estimators = 80, max_depth = 3, learning_rate = 0.001182925709827524 |
| Random Forest (RF) | n_estimators = 87, max_depth = 4, min_samples_split = 9, min_samples_leaf = 9, max_features = 0.793102879894246 |
| Extra Trees (ET) | max_depth = 24, min_samples_split = 19, min_samples_leaf = 1 |
| Deep Neural Network (DNN) | dense_layers = [64, 256, 64, 64], activation = "relu",dropout_rate = 0.2, output_activation = "sigmoid", learning_rate = 0.001, loss = "binary_crossentropy", batch_size = 256 |

### 2.3 Experimenting with Preprocessing Techniques

Preprocessing is crucial for enhancing model efficiency and accuracy by transforming datasets for classification tasks. We experimented with common preprocessing techniques and analyzed the effects of each technique on model performance and training time. Below is a list of various preprocessing techniques we experimented with:

- **Dataset Size Reduction:** We reduced the BENIGN class in CICIDS2017 by 90%, retaining only 10% of benign traffic, while leaving other attack classes unaffected.
- **SMOTE Oversampling:** We addressed class imbalance by generating synthetic samples for minority classes, matching the size of the largest class (BENIGN).
- **Class Grouping:** We grouped minority and similar attack classes to simplify classification complexity.
- **Dataset Scaling:** We scaled the dataset with StandardScaler.
- **Feature Reduction:** We selected the most relevant features to enhance efficiency and accuracy. Using a method similar to [10], we extracted feature importance scores from four tree-based models (Decision Tree, Random Forest, Extra Trees, and XGBoost) and selected the top features for each training dataset.

### 2.4 Capturing and Evaluating Network Traffic

This section describes the methodology for capturing and evaluating network traffic from a local area network (LAN) and analyzing it using ML models. We collect two sets of packets: benign traffic and attack traffic. From each set of capture, we calculated the features using the CICFlowMeter tool [11]. This tool generates a dataset with the same features as CICIDS2017 for evaluation purposes.

**Benign Traffic**

Benign traffic is collected from a LAN with 22 devices performing normal activities like web browsing, video conferencing, web server and SQL server queries, remote desktop, and SSH sessions. Traffic is captured using Wireshark on a capture PC connected to a switch with Port Mirroring enabled. We captured 16 hours of benign traffic: 8 hours for LBP data in the IDS-CICIDS2017+LBP dataset and 8 hours for the testing dataset LAN8h-Combined.

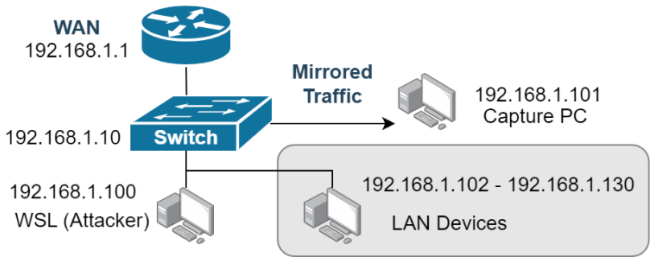


**Fig. 2.** Network layout diagram.

**Attack Traffic**

Capturing attack traffic involves isolating it from benign traffic, complicated by modern OS background activities. To address this, we use Windows Subsystem for Linux (WSL) to run a minimal Kali Linux setup on a Windows PC (192.168.1.109), with its own virtual ethernet adapter and network address translation (NAT). This provides an isolated environment free from background noise, simplifying traffic labeling. We use tcpdump within Kali Linux to capture attack scenarios, including reconnaissance, denial-of-service (DoS), and brute force attempts. The attacking applications we use are identical to the ones used by the authors in CICIDS2017 dataset.

### 2.5 IDS Models

In this section, we detail the steps that we used to create two IDS versions (i.e. two sets of IDS were created):

1. IDS-CICIDS2017: All nine ML models will be trained using only the CICIDS2017 dataset. This acts as a control.
2. IDS-CICIDS2017+LBP (Local Baseline Profile): This version includes the same dataset and techniques but also incorporates the Local Baseline Profile (8 hours of local benign traffic) during training to improve accuracy.

We propose that combining the CIICDS2017 dataset with an LBP should help the IDS adapt to specific network environments. This allows the IDS to generalize better across different network conditions, making it more robust against attacks.

### 2.6 Evaluation Metrics

To evaluate model performance, we use several metrics focusing on the confusion matrix, multiclass average, weighted average precision, recall, and F1-score.

The confusion matrix shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class. It visualizes model performance by comparing predicted classifications to actual classifications.

Weighted-Averaged Metric calculates the metric for each class and then takes the weighted average, where the weight is the number of instances for each class. This approach accounts for class distribution, giving more importance to larger classes.

# 3 Results and Discussion

## 3.1 Impact of Preprocessing

In this section, we analyze the impact of various preprocessing techniques on the performance of intrusion detection systems (IDS) using machine learning (ML) models. These techniques include reducing dataset size, SMOTE oversampling, grouping minority and similar attack classes, dataset scaling, and feature reduction.

Table 2 outlines the various preprocessing stages applied to the CICIDS2017 dataset. Each preprocessing stage builds upon the previous one (except for SMOTE), progressively transforming the dataset to evaluate how these techniques impact the performance and training times of IDS models.

**Table 2.** Preprocessing stages.

| Stage | Preprocessing details |
|---|---|
| None | CICIDS2017 (Original) |
| Downsized | CICIDS2017, Downsized |
| SMOTE | CICIDS2017, Downsized + SMOTE |
| Group Class | CICIDS2017, Downsized + Group Class |
| Scaling | CICIDS2017, Downsized + Group Class + Scaling |
| Reduced Features | CICIDS2017, Downsized + Group Class + Scaling + Reduced Features |

**Reducing Dataset Size.**
We first explored the impact of reducing the dataset size by scaling down the majority classes by a factor of 0.1. we observed that tree-based models (XGBoost, DT, ET, RF) maintained high accuracy (>99%) even with the reduced dataset size. KNN, however, experienced a minor (~2%) reduction in accuracy. Training time was significantly reduced by 90.46%, totaling a reduction of 32,048 seconds. Models like LR, LSVC, and RF greatly benefited from this reduction, with training times decreasing by 17,583, 7,728, and 5,737 seconds respectively. Overall, reducing the dataset size substantially decreased training times with minimal impact on performance for most models, especially tree-based ones. However, LSVC and LR were adversely affected by the reduced training data. It shows that for specific ML models, reducing dataset can give a very attractive tradeoff between accuracy and training times.

**SMOTE Oversampling.**
Next, we assessed the impact of SMOTE (Synthetic Minority Over-sampling Technique) on model performance. Performance differences for top models (XGBoost, KNN, DT, ET, RF) were minimal with less than 1% change. Training time increased by 13,515 seconds due to the enlarged dataset size after balancing. SMOTE did not

enhance overall performance and instead slightly reduced it while increasing training times. LR and LSVC were the most impacted.

**Grouping Minority and Similar Attack Classes.**
We also explored the effect of grouping minority classes and similar attack categories. Top models (XGBoost, KNN, DT, ET, RF) showed minimal performance changes (<1% difference), while LSVC experienced a 6% performance degradation. Grouping classes simplified classification by focusing on broader patterns rather than minute details, reducing training time, especially for LR and LSVC. Training time was further reduced by 2,078 seconds.

**Table 3.** Performance metrics for each preprocessing stage.

| | Accuracy | | | | | | Training Time (seconds) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | None | Downsized | SMOTE | Group Classes | Scaling | Reduced Features | None | Downsized | SMOTE | Group Classes | Scaling | Reduced Features |
| XGB | 99.90% | 99.52% | 99.41% | 99.88% | 99.88% | 99.88% | 217.89 | 27.85 | 195.47 | 11.49 | 10.30 | 8.84 |
| KNN | 99.37% | 97.55% | 96.98% | 98.10% | 99.04% | 98.76% | 0.96 | 0.13 | 0.85 | 0.08 | 0.02 | 0.02 |
| GNB | 66.63% | 73.84% | 72.80% | 74.41% | 77.59% | 77.75% | 5.17 | 0.45 | 4.49 | 0.42 | 0.26 | 0.20 |
| DT | 99.76% | 99.39% | 98.96% | 99.77% | 99.76% | 99.76% | 186.76 | 9.46 | 106.36 | 8.66 | 7.79 | 4.71 |
| ET | 99.19% | 98.68% | 97.34% | 98.99% | 98.91% | 98.98% | 511.48 | 33.18 | 293.26 | 28.54 | 37.79 | 32.11 |
| DNN | 80.31% | 55.20% | 7.91% | 55.20% | 97.57% | 97.13% | 227.36 | 78.62 | 266.01 | 53.56 | 40.31 | 39.67 |
| RF | 99.77% | 99.46% | 99.00% | 99.81% | 99.80% | 99.81% | 6072.59 | 335.62 | 3584.89 | 264.66 | 234.27 | 161.96 |
| LSVC | 69.36% | 65.33% | 62.55% | 59.74% | 76.03% | 70.34% | 8238.95 | 510.78 | 5434.66 | 234.42 | 12.81 | 9.73 |
| LR | 90.87% | 83.62% | 57.66% | 83.49% | 93.95% | 91.49% | 19967.13 | 2383.77 | 7008.40 | 699.78 | 451.23 | 228.08 |

**Dataset Scaling.**
Finally, we analyzed the impact of scaling the dataset StandardScaler. Scaling improved DNN performance, achieving 40% increase. LR benefited too with a 10% increase in performance, GNB saw a 3% increase, and LSVC improved by 12% with StandardScaler. Scaling improved model performance, especially for those relying on normally distributed data (DNN, LR, GNB). It also further reduced training times by 506 seconds.

**Feature Reduction.**
Finally, we evaluated the impact of reducing the number of features to 40. Top models (XGBoost, KNN, DT, ET, RF) showed negligible performance changes (<1% difference). LSVC declined by 6% and LR's performance dropped by ~2%. Most models maintained their performance with fewer features, but GNB, LSVC, and LR showed slight degradation. Training times were generally reduced, particularly for RF and LR.

### 3.2 Real-world Performance of IDS

As discussed in the methodology section, we created two versions of IDS namely, CICIDS2017 and CICIDS2017+LBP (details can be found in Section 2.5).

Then, from both datasets, we created another two variants by reducing their features from 77 to 40. In total, 4 IDS variants were created, details are given below.

**LAN8h-Combined**

We created a dataset named LAN8h-Combined where we captured both normal and malicious traffic from a local area network and transformed it using CICFlowMeter. The aim is to evaluate model performance when subjected to mixed traffic. The dataset is categorized into four classes based on the type of traffic:

**Table 4.** Dataset information – LAN8h-Combined

| Class | Application/Source | Count |
|---|---|---|
| BENIGN | Normal network traffic (8 hours) | 109,433 |
| DoS | Goldeneye, Slowhttptest | 138,137 |
| BruteForce | Patator (SSH and FTP) | 126,221 |
| PortScan | Nmap | 323,309 |

The testing dataset contains 15.7% of benign traffic. Benign traffic contains 8 hours of normal network traffic captured locally. The application used for generating attack traffic (DoS, BruteForce, and PortScan) is identical to the CICIDS2017 dataset.

**Table 5.** Testing results – LAN8h-Combined.

| | IDS-CICIDS2017, 77 features | | | | IDS-CICIDS2017+LBP, 77 features | | | | IDS-CICIDS2017, 40 features | | | | IDS-CICIDS2017+LBP, 40 features | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Accuracy | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 | Accuracy | Precision | Recall | F1 |
| XGB | 15.9% | 20.6% | 15.9% | 4.7% | 15.7% | 2.5% | 15.7% | 4.3% | 15.9% | 20.5% | 15.9% | 4.7% | 15.7% | 2.5% | 15.7% | 4.3% |
| KNN | 16.4% | 10.5% | 16.4% | 6.3% | 15.7% | 2.8% | 15.7% | 4.3% | 15.4% | 8.1% | 15.4% | 4.7% | 16.2% | 22.3% | 16.2% | 5.3% |
| GNB | 15.5% | 2.6% | 15.5% | 4.4% | 15.6% | 2.7% | 15.6% | 4.6% | 15.4% | 2.5% | 15.4% | 4.2% | 15.4% | 2.5% | 15.4% | 4.2% |
| DT | 21.7% | 67.5% | 21.7% | 13.7% | 33.8% | 21.4% | 33.8% | 23.1% | 15.4% | 3.1% | 15.4% | 4.6% | 15.7% | 2.5% | 15.7% | 4.3% |
| ET | 15.7% | 2.5% | 15.7% | 4.3% | 15.7% | 2.5% | 15.7% | 4.3% | 15.7% | 2.5% | 15.7% | 4.3% | 15.7% | 2.5% | 15.7% | 4.3% |
| DNN | 16.9% | 14.7% | 16.9% | 8.8% | 15.7% | 2.5% | 15.7% | 4.3% | 15.4% | 3.2% | 15.4% | 4.4% | 15.7% | 21.5% | 15.7% | 4.3% |
| RF | 15.7% | 20.7% | 15.7% | 4.4% | 33.2% | 21.1% | 33.2% | 22.8% | 15.7% | 2.6% | 15.7% | 4.4% | 15.7% | 2.5% | 15.7% | 4.3% |
| LSVC | 17.2% | 27.8% | 17.2% | 9.0% | 16.0% | 18.9% | 16.0% | 4.9% | 16.9% | 19.8% | 16.9% | 7.0% | 16.3% | 21.6% | 16.3% | 5.4% |
| LR | 15.3% | 6.3% | 15.3% | 4.2% | 15.7% | 2.5% | 15.7% | 4.3% | 16.0% | 48.4% | 16.0% | 6.3% | 15.7% | 2.5% | 15.7% | 4.3% |

Table 5 presents the performance of various ML models trained on CICIDS2017 dataset with varying amounts of features and LBP data, tested against LAN8h-Combined. Most models have low overall accuracy, with the majority around 15-17%. These models correctly identified benign traffic but failed to identify attack traffic. Most of the attack traffic was classified as benign.

Reducing the number of features from 77 to 40 significantly impacted model performance. Both Decision Tree and Random Forest models showed decreased accuracy, dropping to 15.7% accuracy. This suggests that additional features, especially from LBP data, are crucial for distinguishing between attacks like BruteForce, DoS, and PortScan. While feature reduction simplifies the model and reduces costs, it compromises accuracy and robustness in identifying complex attack patterns.

DT and RF models trained on IDS-CICIDS2017+LBP (77 features) and DT trained on IDS-CICIDS2017 (77 features) show significantly higher accuracy (33.8%, 33.2%,

and 21.7% respectively) than others. These three models were able to correctly classify benign and BruteForce attacks but failed to detect other attack types.

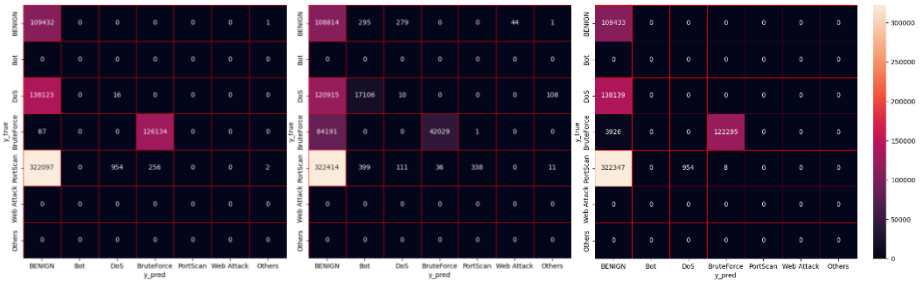


**Fig. 3.** Confusion Matrix, DT trained on IDS-CICIDS2017+LBP, 77 features (left), DT trained on IDS-CICIDS2017, 77 features (center), RF trained on IDS-CICIDS2017+LPB, 77 features (right).

Both DT and RF, trained on IDS-CICIDS2017+LBP with 77 features show similar performance. Fig. 3 shows both models achieve perfect recall for benign instances but have low precision, often misclassifying attack instances as benign. They struggle to correctly identify DoS and PortScan instances. The DT achieves perfect precision for BruteForce but has low recall, while the RF shows nearly perfect precision and recall for BruteForce. In contrast, the DT classifier trained without LBP data show poorer performance when classifying BruteForce attack. This indicates that the proposed method of including LBP data greatly improved DT and RF detection rates for BruteForce attacks specifically. However, there was minimal effect for other attacks.

LR trained on IDS-CICIDS2017 with 40 features scored highest precision (48.4%). However, it has low recall (16%) meaning that while the detection rates are low, the detected attacks were accurate. However, the overall accuracy is still poor (16%).

The CICIDS2017 dataset has been extensively utilized; however, it falls short in capturing attached traffic when incorporating new real-world data. This trend is evident across numerous ML algorithms as shown in Table 4. While the real-world data increased detection rates for certain models like DT and RF for BruteForce category, the overall performance is still poor. This underscores the necessity for more dynamic datasets to accurately classify traffic and enhance the efficiency of IDS systems.

## 4 Conclusion and Future Work

This study presents comprehensive experiments with various preprocessing techniques and evaluates IDS model performance in realistic environments, offering valuable insights for developing more robust and adaptive IDS. We analyzed the impact of preprocessing methods on training times and accuracy. Reducing dataset size, grouping minority classes, and scaling significantly decreased training times while maintaining high accuracy. However, SMOTE oversampling increased training times and degraded performance for some models.

We also assessed IDS performance with different preprocessing techniques in real-world scenarios. Most IDS variants struggled to reliably detect attacks, with high precision but low recall. This highlights the need for more sophisticated datasets with diverse attacks and adaptive IDS capable of evolving with network environments.

Incorporating Local Baseline Profile (LBP) improved detection rates for specific attack vectors using certain models. However, the overall performance of the IDS is still unsatisfactory. Additionally, reducing features negatively impacted real-world performance, contrasting with our experimental findings. This signifies the need to create an improved dataset that enhances the anomaly detection in real-world traffic, and to develop a more sophisticated IDS framework that is adaptive to ever-changing network environment and capable of detecting anomalies reliably and accurately.

Future research should focus on creating more diverse datasets, developing advanced IDS that adapt to evolving networks, and exploring hybrid approaches combining multi-class and binary classification. Experimenting with various preprocessing techniques' real-world impacts is also crucial.

## References


1. J. Asharf *et al.*, "A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions," *Electronics*, vol. 9, no. 7, p. 1177, 2020.
2. A. Petrosyan, "Monthly number of IOT attacks global 2022," Statista, https://www.statista.com/statistics/1322216/worldwide-internet-of-things-attacks
3. A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, 2019.
4. H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.
5. N. Thapa, Z. Liu, D. B. KC, B. Gokaraju, and K. Roy, "Comparison of machine learning and deep learning models for network intrusion detection systems," *Future Internet*, vol. 12, no. 10, p. 167, 2020.
6. R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering & Technology*, 7(3.24), pp. 479-482, 2018
7. A. Thakkar, R. Lohiya, "A Review of the Advancement in Intrusion Detection Datasets," *Procedia Computer Science*, 2020, vol. 167, pp. 636–645.
8. K. A. Binsaeed and A. M. Hafez, "Enhancing Intrusion Detection Systems with XGBoost Feature Selection and Deep Learning Approaches," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 5, Jan. 2023.
9. I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018.
10. L. Yang, A. Moubayed, I. Hamieh and A. Shami, "Tree-Based Intelligent Intrusion Detection System in Internet of Vehicles," *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1-6.
11. A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, 2017, pp. 253–262.