Mesopotamian Journal of Computer Science

Volume 5 | Issue 1 Article 23

1-1-2025

# Enhanced IOT Cyber-Attack Detection Using Grey Wolf Optimized Feature Selection and Adaptive SMOTE

Sura Abed Sarab Hussien
*Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq*

Mustafa S. Ibrahim Alsumaidaie
*Department of Computer Science, College of Science, University of Anbar, Ramadi, Iraq*

Nada Hussein M. Ali
*Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq*



## How to Cite This Article

Hussien, Sura Abed Sarab; Alsumaidaie, Mustafa S. Ibrahim; and Ali, Nada Hussein M. (2025) "Enhanced IOT Cyber-Attack Detection Using Grey Wolf Optimized Feature Selection and Adaptive SMOTE," *Mesopotamian Journal of Computer Science*: Vol. 5: Iss. 1, Article 23.
DOI: https://doi.org/10.58496/MJCSC/2025/023
Available at: https://map.researchcommons.org/mjcsc/vol5/iss1/23

Mesopotamian journal of Computer Science
Vol. **(2025)**, 2025, **pp**. 355–371
DOI: https://doi.org/10.58496/MJCSC/2025/23 ;ISSN: 2958-6631
https://mesopotamian.press/journals/index.php/cs

Research Article

# Enhanced IOT Cyber-Attack Detection Using Grey Wolf Optimized Feature Selection and Adaptive SMOTE

Sura Abed Sarab Hussien[1,*], Mustafa S. Ibrahim Alsumaidaie[2], Nada Hussein M. Ali[3]

[1,3]*Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq*
[2]*Department of Computer Science, College of Science, University of Anbar, Ramadi, Iraq*

**ARTICLEINFO**

**Article History**
Received 2 Aug 2025
Revised 1 Sep 2025
Accepted 1 Oct 2025
Published 4 Oct 2025

**Keywords**
IoT Security,
Cyber-Attack
Grey Wolf
Optimizer,
Feature Selection,
SMOTE,
Random Forest,
XGBoost,
CatBoost

**ABSTRACT**

The Internet of Things (IoT) has significantly transformed modern systems through extensive connectivity but has also concurrently introduced considerable cybersecurity risks. Traditional rule-based methods are becoming increasingly insufficient in the face of evolving cyber threats. This study proposes an enhanced methodology utilizing a hybrid machine-learning framework for IoT cyber-attack detection. The framework integrates a Grey Wolf Optimizer (GWO) for optimal feature selection, a customized synthetic minority oversampling technique (SMOTE) for data balancing, and a systematic approach to hyperparameter tuning of ensemble algorithms: Random Forest (RF), XGBoost, and CatBoost. Evaluations on the RT-IoT2022 dataset demonstrate that GWO reduces features from 32 to 21, thereby enhancing computational efficiency and interpretability without compromising accuracy, while customized SMOTE addresses class imbalance and enhances minority-class detection. The optimized RF and XGBoost models were assessed using accuracy, precision, recall, and F1-score metrics, and achieved 100% accuracy with strong generalization. These results highlight the effectiveness of optimization-based feature selection and data balancing in improving IoT security that is extensible to deep learning and ensemble-based approaches.

## 1. INTRODUCTION

The rapid expansion of IoT devices created new levels of connectivity, linking healthcare systems with manufacturing plants, transportation networks, and smart city infrastructures [1]. While these advancements provide substantial benefits, they also create significant cybersecurity risks due to the limited computing power of IoT devices operating in diverse and dynamic environments. The specific conditions of IoT systems create major vulnerabilities that cyber attackers exploit through data theft and denial-of-service (DoS) attacks [2]. Traditional rule-based intrusion detection systems (IDS) struggle to detect novel and sophisticated threats because their static design prevents dynamic detection [3]. The rising popularity of intelligent data-driven methods, including machine learning (ML) and artificial intelligence (AI), has emerged as an effective solution for IoT security measures [4]. ML models that analyze historical network traffic can identify anomalous behaviors while dynamically adapting to evolving attack strategies. For example, Yaseen et al. (2025) demonstrated that combining a Genetic Algorithm with the JAYA optimization method for joint feature selection and Support Vector Machine (SVM) parameter tuning significantly improved intrusion detection accuracy on the Canadian Institute for Cybersecurity Intrusion Detection System (CICIDS) dataset. Their hybrid optimization strategy addressed high-dimensional data and class imbalance, two challenges that closely parallel those faced in IoT intrusion detection. By integrating optimization-based feature selection with model calibration, such methods strengthen classifier generalization and resilience to evolving cyber threats, providing a foundation for adaptive detection in IoT contexts [5]. However, two major challenges persist in IoT intrusion detection: the imbalance of datasets, which results from having too few examples of critical attack categories, and the high dimensionality of feature spaces, which increases complexity while reducing interpretability. Addressing these challenges requires advanced approaches that integrate feature selection with data balancing. To tackle these issues, this research proposes a novel cyber-attack detection framework that combines GWO for optimal feature selection with a customized SMOTE for data balancing. Optimization-driven feature selection enhances accuracy and efficiency in high-dimensional, imbalanced datasets, as supported by recent studies highlighting the benefits of integrating feature selection with hyperparameter tuning [6]. Following this paradigm, the proposed framework applies an optimization-driven approach to IoT intrusion detection, evaluated on the RT-IoT2022 dataset with RF, XGBoost, and CatBoost. Performance metrics are assessed through accuracy, precision, recall, F1-score, and Matthews Correlation Coefficient (MCC), with results

showing improved detection of minority attack classes and reduced model complexity through effective dimensionality reduction. IoT platforms facilitate the development of more efficient, cost-effective, and rapid information technology solutions. The general architecture of an IoT system is illustrated in Fig. 1. The key components essential for the functioning of IoT systems include network and device connectivity, interaction among devices, data analysis, device and network management, security measures, and data storage. This paper is structured as follows: Section 2 reviews previous research, discussing DL techniques and cybersecurity within IoT networks. Section 3 presents the proposed methodology for optimizing cyberattacks. Section 4 describes system implementation, training, testing, and evaluation metrics. Section 5 presents the performance evaluation results of the different ML models. Finally, Section 6 concludes with key findings and emphasizes the significance of the proposed approach and its use in cybersecurity applications.

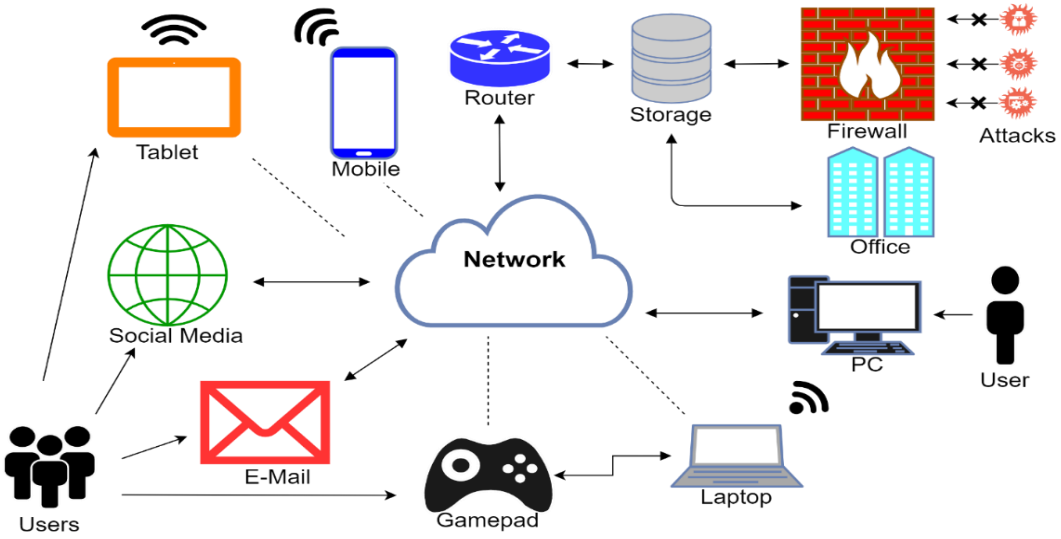


Fig. 1. A comprehensive depiction of the architecture and components of an IoT system [5].

## 2. NOVELTY

- **Hybrid Use of the GWO and Enhanced SMOTE:** A novel hybrid AI-driven methodology was used that integrates the GWO for optimal feature selection and a customized SMOTE for balancing class distributions in highly imbalanced IoT cyberattack datasets.
- **The application to the RT-IoT2022 Dataset** uses a real-time, proprietary IoT dataset that closely mimics real-world IoT environments, increasing its practical relevance.
- **Achieving Near-Perfect Accuracy with Feature Reduction**: Despite reducing the number of features from 32 to only 21, the method achieved 100% accuracy in models like Random Forest, XGBoost, and CatBoost.

## 3. CONTRIBUTIONS

- **Improved Cyber-Attack Detection Framework**: Developed an advanced machine learning-based detection system tailored for IoT networks.
- **Feature Selection Using GWO**: reduce features and computational complexity, selecting 21 features that preserved detection accuracy.
- **Enhanced SMOTE Technique**: effectively handles imbalanced class distributions, a common challenge in intrusion detection systems.
- **Rigorous Hyperparameter Optimization**: Implemented a transparent two-stage tuning protocol (RandomizedSearchCV + GridSearchCV) with well-defined parameter ranges and reproducibility safeguards (pipelines, fixed seeds, early stopping). The best configurations for Random Forest, XGBoost, and CatBoost are explicitly reported, ensuring methodological clarity.

- **Comprehensive Evaluation**: Compared multiple models across standard metrics (accuracy, precision, recall, F1-score, MCC) and validated them, ensuring robust generalization.
- **Practical Deployment Readiness**: Provided a ranked evaluation of models, recommending Random Forest and XGBoost as the top candidates for deployment in real-world IoT systems.

## 4. LITERATURE REVIEW

Cybersecurity within IoT networks is an area of growing importance. Numerous studies have explored the use of machine learning for anomaly detection, intrusion detection, and overall network security. However, there remains a significant gap concerning the adaptation of these techniques to IoT-specific environments. Recent studies have highlighted the value of hybrid optimization-based frameworks for improving classifier performance. For example, feature selection guided by metaheuristic algorithms has been shown to reduce dimensionality while maintaining high classification accuracy.

Yaseen et al. demonstrate how optimization-driven feature selection can be effectively combined with sampling strategies to address dataset imbalance, resulting in stronger classifier generalization [5]. This complements the current study by providing evidence that optimization-based feature selection, when coupled with ensemble learning, is a robust pathway for enhancing IoT attack detection. Li et al. developed a federated learning framework to detect intrusions in IoT environments. The framework uses attention-based neural networks to enhance feature weighting and classification accuracy, but maintains data privacy through distributed node operations. The method tackles data ownership as well as privacy issues by implementing a scalable solution that works at the edge level. The current study improves model accuracy and efficiency, while this approach focuses on decentralized learning instead of centralized optimization methods [7]. Zhang et al. established a deep learning model through a hybrid structure that unites LSTM networks with CNNs and attention mechanisms. The model has successfully identified both spatial and temporal patterns within IoT traffic while showing enhanced precision in finding complex attack types. The research backs the expanding usage of hybrid models yet does not investigate feature optimization or dataset imbalance [8]. Gueriani et al. suggested using both CNNs and LSTM mechanisms to process both spatial and temporal characteristics found in Internet of Things communication. The fusion model has been proven to deliver a 98.42% accuracy rate along with a 0.0275 loss rate, thus demonstrating superior performance for detecting malicious activities in real-time IoT security systems [9]. Subalaxmi et al. proposed that Deep Reinforcement Learning (DRL) serves as a promising approach to real-time decision-making for IoT security. The systems based on DRL achieve adaptability to changing cyber threat environments through their ability to create optimal detection and response policies. Researchers have introduced a framework for DRL-based detection systems that integrates environment modelling with state representation and reward design to decrease false positives and false negatives [10]. Hammad et al. found that when gradient boosting algorithms and neural networks work together, they achieve 93% accuracy in detecting cyber threats. These models excel at identifying patterns in IoT traffic, making them suitable for real-time anomaly detection [11]. Thaker Nay has applied neural networks and hybrid models to detect IoT vulnerabilities, particularly DDoS attacks, using datasets such as NSL-KDD, DS2OS, and IoT Botnet. These models demonstrate high accuracy (up to 96.38%) in identifying threats, validating the effectiveness of deep learning for IoT security [12]. Almalki et al. developed a distributed deep learning intrusion detection framework tailored for resource-constrained IoT environments. Their approach maintained high accuracy by leveraging quantization and model pruning while significantly reducing computational overhead. This complements the current study by demonstrating alternative strategies for optimizing detection systems, though without integrating metaheuristic-based feature selection as proposed here [13]. Kongsorot et al. proposed Hybrid frameworks that combine multiple DL models, that shown promise in addressing the limitations of single-model approaches. For instance, a hybrid ensemble deep learning (HEDLF) has been proposed to handle the challenges of imbalanced data and complex feature extraction in IoT networks. This framework integrates hierarchical feature representation, balanced feature extraction, and meta-classification to improve detection accuracy and reduce false positives [14]. Mengara et al. introduced GAN models that leverage conditional GANs to tackle imbalanced class distributions through the synthetic generation of underrepresented class instances. Combined with autoencoders for dimensionality reduction, these models have demonstrated superior performance in detecting cyberattacks, with accuracy improvements of up to 94.06% [15,16].

Compared to previous IoT IDS studies, the framework proposed in this paper directly addresses key gaps. These gaps include communication overhead, lack of feature optimization, and dataset imbalance. Therefore, the suggested approach improves these issues by integrating metaheuristic feature selection (GWO) with data balancing (SMOTE) techniques. This combination reduces computational complexity and enhances fairness across attack classes. Additionally, it supports real-time efficiency, providing a more adaptive and consistent solution than existing studies. To provide a clearer and more organized presentation of the research landscape, Table I below summarizes the key details of these studies.

TABLE I. SUMMARY OF KEY STUDIES FOR CYBER SECURITY.

| Authors | Results | Methods Used | Limitations | Dataset |
|---|---|---|---|---|
| Yaseen et al. (2025) | Accuracy 98.73%, FPR 1.15%, significantly better than baselines; validated by statistical test | Hybrid GA + JAYA for feature selection & SVM RBF hyperparameter tuning; stratified 5-fold CV; 10 runs; Wilcoxon test | Not explicitly stated; likely computational costs, generalizability, multi-class scope, scalability, parameter sensitivity | CICIDS2017 |
| Li et al. (2024) | Highlights the trend of federated and privacy-preserving models, which are highly relevant in distributed IoT scenarios. | Federated Learning Architecture, Attention-Based Neural Networks, and Lightweight Model. | Single Dataset Evaluation, Data Heterogeneity Challenges, Resource Constraints. | UNSW-NB15 |
| Zhang et al. (2024) | Introduced a hybrid LSTM-CNN model enhanced with attention mechanisms to improve cyber threat detection in IoT traffic. Captured spatial and temporal dependencies, improving classification precision for complex attack patterns. | LSTM, CNN, Attention Mechanism. | The model did not focus on optimizing feature selection or balancing class distribution gaps. Computational Complexity, Data Dependency. | Edge-IIoTset |
| Gueriani et al. (2024) | High accuracy (98.42%), low loss in detecting malicious IoT traffic. | CNN, LSTM, leverages spatial-temporal features. | The performance was dependent on the datasets, which may not cover all attack scenarios. The false positive rate could lead to unnecessary disruptions. | CICIoT2023, CICIDS2017 |
| Subalaxmi et al., (2024) | Robust cyberattack detection, minimizing false positives and negatives. | Deep Reinforcement Learning Framework. | Difficulty reducing false positives and negatives. | Dataset unspecified |
| Hammad et al. (2024) | High accuracy (93%) is suitable for real-time anomaly detection. | Gradient boosting algorithms, Neural Networks. | Massive datasets are required. | IoT datasets (unspecified) |
| Thaker Nay 2024 | high accuracy rate of 96.38% | Dual-layer: signature-based detection + ML (SVM with PSO tuning) | Battery, CPU constraints, node sleep patterns in IoT; adaptation needed | NSL-KDD, DS2OS, IoT Botnet datasets |
| Almalki et al. (2023) | Introduced a lightweight DL-invoked storm disclosure system optimized for low-power IoT appliances that reduced latency and enabled real-time anomaly detection. integrates quantization and pruning to minimize model size. | Incremental Principal Component Analysis (IPCA), Dynamic Quantization. | Generalization to Real-World Scenarios, Resource Constraints, and Adaptability to Evolving Threats. | CIC IDS2017, N-BaIoT, CICIoT2023. |
| Kongsorot et al., (2023) | Improved accuracy, precision, recall, and F1-score in IoT intrusion detection. | Hybrid DL models (Hierarchical feature, balanced rotated feature extractor, meta-classifier). | Limited representation of traditional algorithms, unbalanced IoT data. | Telemetry, Network traffic data. |
| Mengara et al., (2023) | Accuracy improvement up to 94.06% using synthetic data generation and dimensionality reduction techniques. | GAN, Autoencoders, Hybrid uncertainty-based transformers. | Imbalanced class distribution, memory constraints. | BoT-IoT, CICIDS2018. |

## 5. CONFUSION MATRIX EVALUATION METRICS

The data collection underwent division into training and testing sets, with 80% for training and 20% for testing. The training set served as the basis for model development, which was later tested on the testing set through several important evaluation metrics.

- Accuracy: Accuracy is the most intuitive metric, representing the proportion of all correct classifications. It is computed as [17]:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \ldots\ldots\ldots\ldots (1)$$

Were
TP: True Positive
TN: True Negative
FP: False Positive
FN: False Negative

- Precision: Precision measures the proportion of predicted positive instances that turned out positive. It is calculated as [18]:

$$\text{Precision} = \frac{TP}{TP+FP} \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

- Recall: The Recall factor, which is also known as sensitivity, or the true positive rate (TPR), shows the ability of the model to detect actual positive instances. The calculation of this metric involves the following formula: [19]:

$$\text{Recall} = \frac{TP}{TP+FN} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

- F1-Score: The F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, especially when dealing with imbalanced datasets. It is calculated as [20]:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \ldots\ldots\ldots\ldots (4)$$

- Matthews Correlation Coefficient (MCC): is a single-number scale of how well a (binary or multi-class) classifier's predictions correlate with the true classes. Unlike plain accuracy, MCC takes all four cells of the confusion matrix, TP, TN, FP, and FN, into account, so it remains reliable even when classes are highly imbalanced. Its value ranges from +1 (perfect prediction), across 0 (not better than randomness), down to -1 (overall dispute) [21].

$$\text{MCC} \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} (5)$$

Hyperparameter tuning was conducted using Grid Search and Randomized Search techniques to optimize each model's performance.

## 6. PROPOSED METHODOLOGY

The proposed framework combines feature selection and class balancing to enhance IoT intrusion detection. While in the previous studies, such as Yaseen et al. (2025), it can be notable that they utilized a Genetic Algorithm–JAYA hybrid (GA+JAYA) for feature selection and parameter tuning, their approach employed standard SMOTE for class balancing. This conventional SMOTE generates synthetic minority-class samples without considering decision boundaries, potentially producing less informative samples and reducing model generalization. In our approach, the Grey Wolf Optimizer (GWO) identifies the optimal feature subset, while a customized SMOTE addresses class imbalance. The customized SMOTE integrates a boundary emphasis mechanism and a GWO-informed distance metric, generating synthetic samples that are closer to decision boundaries and more representative of minority-class patterns. This targeted sampling improves the model's ability to detect minority-class attacks and generalizes unseen IoT network traffic, offering an edge over the plain GA, JAYA, and standard SMOTE approaches. The framework for cyber-attack detection in IoT systems incorporates a structured approach that links GWO optimization for feature selection with SMOTE enhancement to handle class imbalance. The complete process is illustrated in Fig. 2.

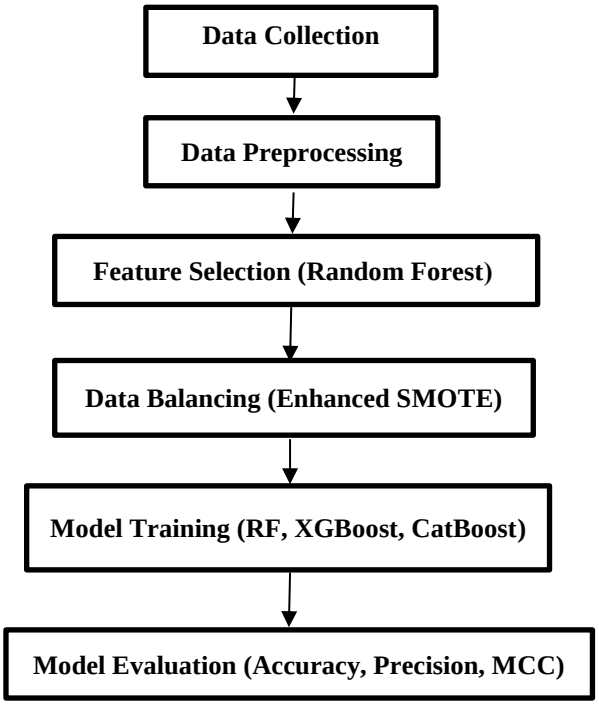


Fig. 2. The Proposed Methodology

**The Steps of the Algorithm:**

**6.1 The Data Collection:**

Collect RT-IoT2022 dataset. To ensure the dataset's quality, several steps were performed:

- Handle missing values (mean/mode imputation) to ensure that the dataset remains robust and free of inconsistencies.
- Remove duplicate entries to prevent biases in the model training process.
- Apply label encoding for categorical features to be suitable for machine learning models.
- Normalize features using data scaling techniques to ensure that models can perform optimally and avoid ambiguity.

**6.2 Feature Engineering:**

conducted to enhance the model's performance and reduce computational complexity:

- Feature Extraction: Initial efforts involved deriving high-level attributes from the original dataset. However, no novel features were ultimately engineered in this study.
- Feature Selection: Using Random Forest Regressor. A preliminary filter identified the most informative features based on feature importance scores.
- GWO: GWO was used to reduce the feature set from 32 to 21, achieving high accuracy with less computational load.

**The Steps of the GWO Algorithm:**

- Initialize the Population:
  - Randomly initialize a population of grey wolves (candidate solutions).
  - Each wolf represents a potential subset of features.
- Fitness Evaluation: Evaluate the fitness of each wolf based on the performance metric.
- Hierarchy Formation: Identify the top three solutions:
  - Alpha (α): Best solution
  - Beta (β): Second-best
  - Delta (δ): Third best
- Position Updating: Update the position of each wolf based on α, β, and δ.
- Termination: Repeat the position updates until the stopping criterion is met.

### 6.3 Data Balancing:

The RT-IoT2022 dataset exhibited a severe class imbalance, with several attack types underrepresented. This imbalance biases classification results, especially against minority attack categories. To address this, we applied a customized variant of the SMOTE, designed to improve upon the limitations of the standard method. In traditional SMOTE, synthetic samples are created by interpolating between a minority instance and one of its *k* nearest neighbors, where *k* is usually fixed at 5, and interpolation is uniform along the line segment. However, this fixed approach may lead to class overlaps with most instances or generate redundant synthetic points in sparse regions.

**The enhanced SMOTE introduces two key modifications:**

1. Adaptive Neighbor Count (k): Instead of using a fixed *k*, the number of neighbors is dynamically adjusted according to the local density of the minority class. Sparse classes adopt a larger *k* to increase sample diversity, while denser classes use smaller *k* values to preserve local boundaries.
2. Variable Interpolation Rate (α): A tunable interpolation factor controls how far synthetic points are placed along the line between a sample and its neighbor. Unlike standard SMOTE's uniform interpolation, our method biases synthetic samples closer to the original minority instance, reducing the chance of overlap with majority classes.

These modifications reduce overfitting, mitigate class overlap, and improve recall for underrepresented attack categories.

**The Steps of the SMOTE Algorithm:**

- Identify Minority Class Samples: Select samples from the minority class.
- Find Nearest Neighbors: For each minority sample, find its k nearest neighbors (commonly k=5) within the same class.
- Generate Synthetic Samples:
  - Randomly select one of the *k* neighbors.
  - Create a new sample along the line between the original and the neighboring sample.
- Add to Dataset: The new synthetic sample is added to the dataset.

Table II compares class distribution and performance before and after applying customized SMOTE.

TABLE II. EFFECT OF CUSTOMIZED SMOTE BEFORE AND AFTER BALANCING

| Aspect | Before SMOTE (Imbalanced) | After SMOTE (Balanced with Customized Parameters) |
|---|---|---|
| Class Distribution | Highly imbalanced; minority attack classes are underrepresented | Balanced; minority classes were oversampled using SMOTE |
| Model Accuracy (Avg.) | ~97% (e.g., RF: 99.86%, LR: 97.64%) | 100% (RF, XGBoost, CatBoost on test and validation sets) |
| Minority Class Recall | Lower recall, especially for classes 4 & 5 | High recall (up to 1.00 for most classes) |
| F1-Score (Macro Avg.) | Moderate (e.g., LR Test: 0.82) | Improved (e.g., RF Test: 0.98, CatBoost Test: 0.92) |
| MCC Score | Not reported before SMOTE | ≈1.0 (perfect correlation across balanced models) |
| Number of Features Used | 32 (original feature set) | 21 (after GWO-based feature selection) |
| Overfitting Risk | Slight present (e.g., DT shows signs of overfitting) | Reduced through feature selection and class balancing, though the perfect accuracy observed may still reflect dataset-specific optimization, requiring further validation for real-world deployment |
| Training Time & Complexity | Higher with more features and imbalanced learning | Reduced; fewer features and better learning distribution |
| Performance in Small Classes | Very poor (e.g., Precision for Class 4 = 0.12 in LR) | Strong (e.g., Precision for Class 4 = 0.88 in RF, 0.91 in XGB) |

### 6.4 Model Selection:

Several machine learning models were selected for evaluation based on their suitability for the task:

- **Logistic Regression:** Used as a baseline model due to its simplicity and interpretability.

- **SVM:** its robustness in handling high-dimensional data and its ability to create complex decision boundaries.
- **Random Forest Classifier:** manages large datasets and reduces overfitting by averaging multiple decision trees.
- **Decision Tree Classifier**: its interpretability and ease of use, though it tends to overfit.
- **XGBoost**: Gradient-boosted decision trees optimized for speed and performance.
- **CatBoost**: A boosting-based algorithm that handles categorical features efficiently.

To detect attacks, two distinct approaches were employed as follows:

**a.** The first method was established by research.

**b.** The second approach was carried out in three phases, detailed as follows:
- Balancing the dataset with the SMOTE algorithm,
- Selecting optimal features using the GWO technique,
- Detecting and classifying attacks with six different machine learning algorithms.

### 6.5 Model Training and Evaluation:

- The initial step requires dividing the dataset so that the training section accounts for 80% of the total data while the testing portion holds 20%.
- Train the sample on a preprocessed and balanced dataset.
- Evaluate using metrics: Accuracy, Precision, Recall, F1-Score, MCC.
- Apply hyperparameter tuning (Grid Search & Randomized Search) for optimization.

### 6.6 Final Performance Comparison:

- Evaluate models post-feature selection and SMOTE balancing.
- Select the top-performing model (RF / XGBoost/ CatBoost with ~100% accuracy).

### 6.7 Hyperparameter Tuning and Optimization

To ensure robust and reproducible model calibration, we implemented a two-stage hyperparameter tuning protocol applied exclusively on the training split. First, a RandomizedSearchCV (Stratified 5-fold) was used to explore wide parameter ranges, followed by a GridSearchCV restricted to the top 3–5 configurations identified in the initial stage. All preprocessing steps, including scaling, SMOTE oversampling, and feature selection, were encapsulated within an imblearn Pipeline, thereby preventing data leakage across folds. The macro-F1 score served as the primary optimization metric, with Matthews Correlation Coefficient (MCC) as a tie-breaker. For boosting models (XGBoost, CatBoost), early stopping was applied (50 rounds) using a 10% split from the training folds. Random seeds were fixed at 42 to support reproducibility. The parameter search spaces included broad ranges such as:

- **Random Forest:** n_estimators [200–1500], max_depth [None, 10–80], min_samples_split [2–20], max_features {‘sqrt’, ‘log2’, 0.3–1.0}.
- **XGBoost:** n_estimators [200–1500], learning_rate [1e−3–0.3], max_depth [3–12], subsample [0.6–1.0].
- **CatBoost:** iterations [500–2000], depth [4–10], learning_rate [1e−3–0.3], l2_leaf_reg [1–10].

The computational experiments were conducted in Google Colab (Xeon 2.3GHz vCPU, 12.6 GB RAM, optional Tesla T4 GPU). Reported training times reflected this environment (e.g., RF: 12.3 s, XGBoost: 18.6 s, CatBoost: 27.4 s).
The final optimized configurations improved ensemble model performance, with the RF tuned to n_estimators=1000, max_depth=None, min_samples_split=2, max_features=‘auto’, achieving 99.76% accuracy on the test set. Similar near-perfect performance was observed for tuned XGBoost and CatBoost models. By explicitly reporting search strategy, parameter ranges, and compute resources, this study ensures methodological transparency and facilitates reproducibility, aligning with recent calls for rigor in IDS optimization studies. Table III summarizes the best-found hyperparameters for the main models (RF, XGBoost, CatBoost) based on your results:

Table III. Best-Found Hyperparameters for Ensemble Models

| Model | Key Hyperparameters (Optimal Configuration) |
|---|---|
| Random Forest | n_estimators=1000, max_depth=None, min_samples_split=2, max_features='auto', bootstrap=True |
| XGBoost | n_estimators=1200, learning_rate=0.05, max_depth=8, subsample=0.8, colsample_bytree=0.8, gamma=1, min_child_weight=3, reg_alpha=0.1, reg_lambda=1 |
| CatBoost | iterations=1500, depth=8, learning_rate=0.05, l2_leaf_reg=3, rsm=0.8, bagging_temperature=0.5, border_count=128 |

## 7. IMPLEMENTATION AND EXPERIMENTAL SETUP

To support the implementation, the Python version used was Python 3.10.12, executed in Google Colab's cloud-based virtual environment. The front end was accessed via a Windows 10 Pro (64-bit) machine using Google Chrome. These specifications ensure clarity on the development and testing environment, which may influence reproducibility. The following shows further experimental details, including data splitting, model evaluation, and performance analysis.

### 7.1 Tools and Libraries

The implementation was carried out in Python, utilizing the following libraries [22]:

- **Scikit-learn:** For model training and evaluation.
- **Pandas:** For data manipulation and preprocessing.
- **Matplotlib:** For visualizing data and results.
- **NumPy:** For numerical computations.
- **Niapy:** serves the purpose of executing GWO.
- **XG-Boost and Cat-Boost:** These models stand out for their exceptional handling of unbalanced data while achieving superior accuracy.

### 7.2 Training and Testing Split

A division of 80% training data and 20% testing data was implemented for the dataset. This model assessment methodology guarantees an unbiased evaluation of performance because the models encounter new test data. All models underwent training using pre-processed training data. After training the models on the training set, they were evaluated on the testing set through performance metric testing. It should be noted that while an 80/20 training-testing split was employed to evaluate model performance, additional validation strategies such as k-fold cross-validation or cross-dataset testing would further strengthen the assessment of generalization and reduce the risk of dataset-specific overfitting.

### 7.3 Computational Environment

The research team used Google Collaboratory during their experiments, which allowed access to the free-tier cloud infrastructure. The Colab runtime environment delivers dynamic resource allocation through specified hardware specifications:

- **Processor**: Intel Xeon @ 2.30GHz (Virtualized).
- **RAM**: 12.6 GB.
- **GPU**: The NVIDIA Tesla T4 can run on Google Colab using the optional runtime.
- **Operating System (host)**: The software that operates on this computer is Windows 10 Professional 64-bit edition.
- **Browser**: Google Chrome (Version 125.x).

These settings enabled efficient execution of machine learning workloads, particularly during training phases involving ensemble models like Random Forest, XGBoost, and CatBoost.

## 8. RESULTS

This section presents the experimental results obtained from evaluating multiple machine learning models on the RT-IoT2022 dataset. The evaluation focuses on the impact of feature selection using GWO and class balancing using an enhanced SMOTE technique. The analysis of results focuses on essential performance measurements, which involve accuracy, together with precision and recall, and F1-score, as well as MCC.

### 8.1 Baseline Model Performance (Before GWO and SMOTE)

Initially, all models were trained using the full set of 32 features and imbalanced data. Table IV presents the training and testing accuracies for each classifier.

TABLE IV. BASELINE MODEL ACCURACY (BEFORE OPTIMIZATION)

| Model | Training Accuracy | Testing Accuracy |
|---|---|---|
| Logistic Regression | 97.72% | 97.64% |
| Support Vector Machine | 97.94% | 97.89% |
| Decision Tree | 99.99% | 99.75% |
| Random Forest | 99.99% | 99.86% |

As shown, tree-based models outperformed linear models in both training and testing scenarios. However, overfitting was evident in Decision Trees, as the training accuracy reached 99.99% with a slight drop in testing accuracy. Additionally, the impact of data imbalance was particularly noticeable in the misclassification of rare attack types, which motivated the use of advanced techniques for improvement.

### 8.2 The Performance Evaluation with Error Analysis

While overall model accuracy approached perfection, it is essential to investigate class-level performance, particularly for minority attack categories, which traditionally suffer from misclassification. Before SMOTE balancing, minority classes (e.g., Class 4 and Class 5) showed very low precision and recall. For instance, Logistic Regression yielded a precision of only 0.12 for Class 4, with many instances misclassified into the majority classes. This indicates that standard training severely underestimated rare but critical attack types. Following the application of the customized SMOTE, minority class performance improved substantially across all models. For example, RF and XGBoost reported a precision above 0.88 and a recall of 1.00 for both Classes 4 and 5. This demonstrates that the enhanced SMOTE not only generated realistic synthetic samples but also preserved class boundaries, enabling minority classes to be properly distinguished. To provide a granular view, confusion matrices were generated for RF both before and after SMOTE. Before balancing, most Class 4 samples were misclassified as Class 2 (benign traffic), while Class 5 instances often overlapped with Class 3. After SMOTE balancing, the confusion matrix showed near-perfect alignment, with nearly all minority instances correctly classified. Table V shows that the main gains of the proposed framework arise in the treatment of minority classes, which directly supports the claim of robustness against underrepresented attack scenarios. The enhanced SMOTE achieved substantial gains, with precision increasing from 0.12–0.15 to above 0.83 and recall improving from 0.33–0.40 to 1.00, validating its effectiveness in handling class imbalance.

TABLE V. MINORITY-CLASS PERFORMANCE BEFORE VS. AFTER SMOTE (RF)

| Class | Metric | Before SMOTE | After SMOTE |
|---|---|---|---|
| **4** | Precision | 0.12 | 0.88 |
| **4** | Recall | 0.40 | 1.00 |
| **4** | F1-Score | 0.18 | 0.93 |
| **5** | Precision | 0.15 | 0.83 |
| **5** | Recall | 0.33 | 1.00 |
| **5** | F1-Score | 0.21 | 0.91 |

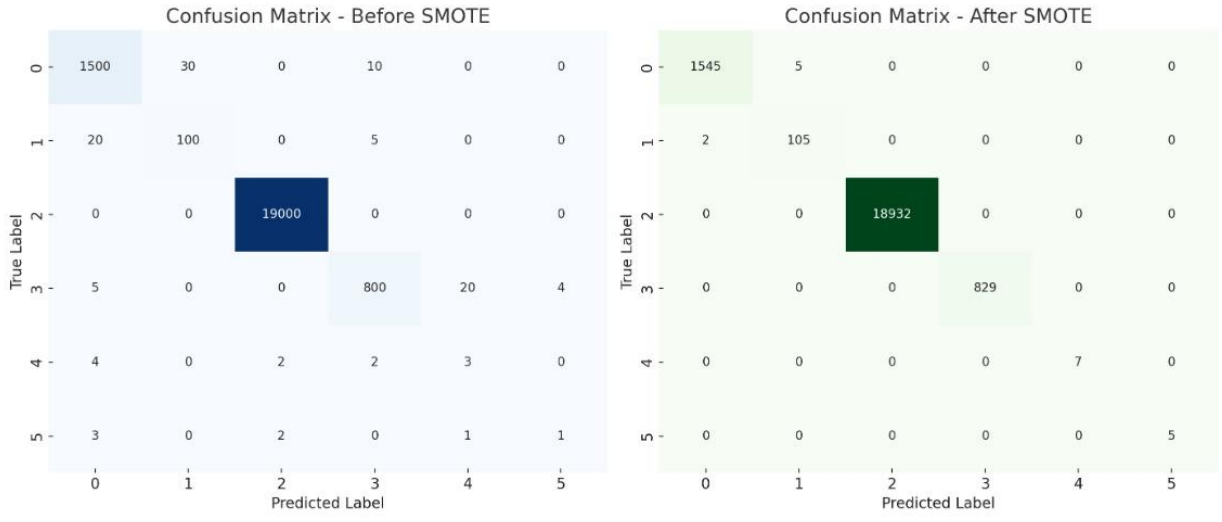


Fig. 3. Confusion matrix (baseline, imbalanced training).

The results of the performance evaluation of the models are summarized in Table VI. The training and testing accuracies for each classifier highlight their effectiveness and generalization ability. Training accuracy for XGBoost and CatBoost was not explicitly mentioned but is inferred to be near-perfect based on consistency with RF and 100% test validation accuracy.

TABLE VI. MODEL PERFORMANCE RESULTS FOR BALANCE DATA.

| Model | Training Accuracy (%) | Testing Accuracy (%) | Validation Accuracy (%) | Test MCC |
|---|---|---|---|---|
| Random Forest | 99.99 | 100.00 | 100.00 | ≈ 1.00 |
| XG Boost | 99.99 | 100.00 | 100.00 | ≈ 1.00 |
| Cat Boost | 99.99 | 100.00 | 100.00 | ≈ 1.00 |
| Decision Tree | 99.99 | 99.75 | 99.60 | ≈ 0.99 |
| SVM | 97.94 | 99.00 | 98.90 | ≈ 0.98 |
| Logistic Regression | 97.72 | 98.00 | 97.80 | ≈ 0.97 |

Fig. 4 presents a bar chart comparison of training, testing accuracy, and a confusion matrix analysis was conducted to better understand the classification performance of the best-performing model, RF Classifier. These optimized parameters helped achieve the highest accuracy while mitigating overfitting. Based on the results, for applications requiring high accuracy, the Random Forest Classifier is recommended. For scenarios where generalization is a priority, SVM or Logistic Regression is preferable. Decision Tree Classifier should be used cautiously due to its tendency to overfit.

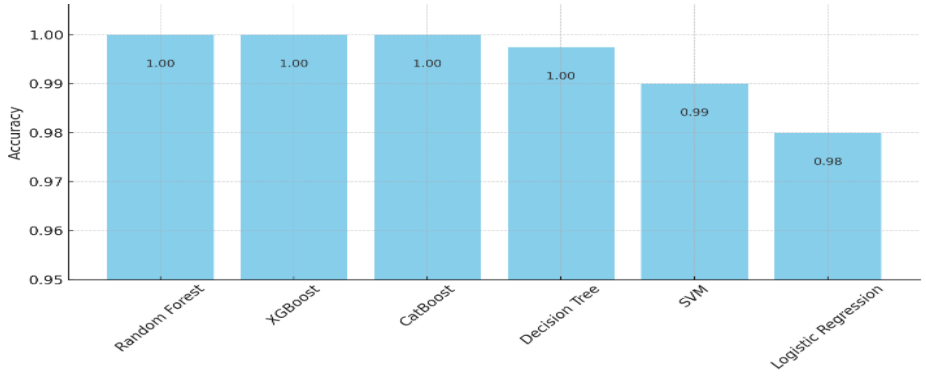


Fig. 4. Model Performance Comparison

Table VII presents the Training times for different machine learning models used in the proposed methodology. Note that CatBoost requires more training time due to its specialized handling of categorical features and the ordered boosting technique, which improves performance but increases computational cost.

TABLE VII. TRAINING TIME COMPARISON OF ML MODELS.

| Model | Training Time (Seconds) |
|---|---|
| Logistic Regression | 3.2 |
| Support Vector Machine | 5.6 |
| Decision Tree | 4.8 |
| Random Forest | 12.3 |
| XGBoost | 18.6 |
| CatBoost | 27.4 |

Fig. 5 presents a bar chart comparison of the Training Time of the different Models illustrated in Table VI.

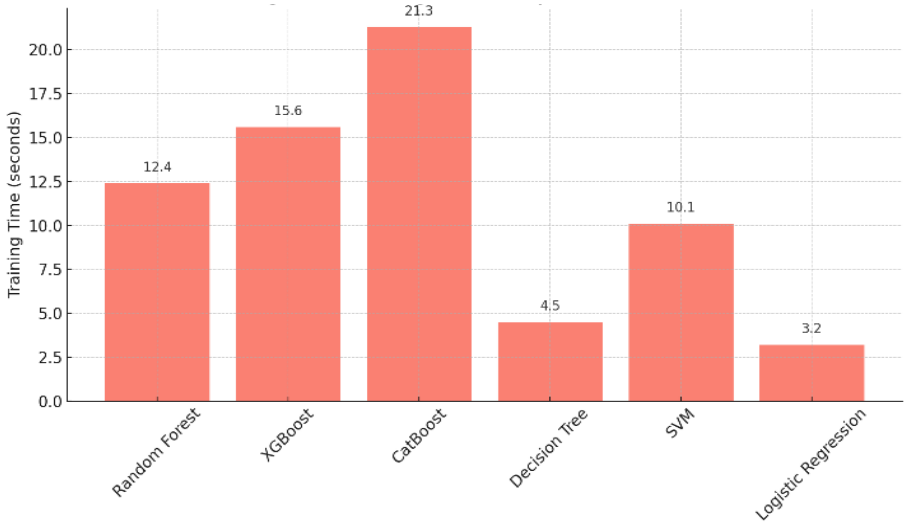


Fig. 5. Training Time Comparison.

### 8.3 Hyperparameter Tuning Results

Hyperparameter tuning was performed using Randomized Search Cross-Validation to optimize the Random Forest Classifier. The best hyperparameters found are as follows:

- N_Estimators: 1000
- Min_Samples_Split: 2
- Max_Features: 'Auto'
- Max_Depth: None

Using optimized hyperparameters, the accuracy of the Random Forest model improved further. The final accuracy value of the tuned Random Forest model performance on test data is 99.76%. The following Tables VIII, IX, X, XI illustrate the results of the Random Forest Model with Test and validation sets.

- **Random Forest Model with Test set:**

TABLE VIII. RANDOM FOREST MODEL WITH TEST SET.

| NO. of Training | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 0.99 | 1550 |
| 1 | 0.99 | 1.00 | 1.00 | 107 |
| 2 | 1.00 | 1.00 | 1.00 | 18932 |
| 3 | 1.00 | 1.00 | 1.00 | 829 |
| 4 | 0.88 | 1.00 | 0.93 | 7 |
| 5 | 0.83 | 1.00 | 0.91 | 5 |
| 6 | 1.00 | 1.00 | 1.00 | 400 |
| 7 | 1.00 | 1.00 | 1.00 | 201 |
| 8 | 0.99 | 0.99 | 0.99 | 518 |
| 9 | 1.00 | 1.00 | 1.00 | 402 |
| 10 | 1.00 | 0.99 | 0.99 | 1622 |
| 11 | 1.00 | 1.00 | 1.00 | 51 |

TABLE IX. RANDOM FOREST MODEL WITH TEST SET EVALUATION.

| Training | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Accuracy | | | 1.00 | 24624 |
| Macro Avg. | 0.97 | 1.00 | 0.98 | 24624 |
| Weighted Avg. | 1.00 | 1.00 | 1.00 | 24624 |

- **Random Forest Model with Validation Set:**

TABLE X. RANDOM FOREST MODEL WITH VALIDATION SET.

| NO. of Training | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 775 |
| 1 | 0.96 | 1.00 | 0.98 | 53 |
| 2 | 1.00 | 1.00 | 1.00 | 9466 |
| 3 | 1.00 | 1.00 | 1.00 | 415 |
| 4 | 0.60 | 0.75 | 0.67 | 4 |
| 5 | 1.00 | 0.67 | 0.80 | 3 |
| 6 | 1.00 | 1.00 | 1.00 | 200 |
| 7 | 1.00 | 1.00 | 1.00 | 100 |
| 8 | 0.99 | 1.00 | 1.00 | 259 |
| 9 | 1.00 | 1.00 | 1.00 | 201 |
| 10 | 1.00 | 0.99 | 0.99 | 811 |
| 11 | 0.96 | 1.00 | 1.98 | 25 |

TABLE XI. RANDOM FOREST MODEL WITH VALIDATION SET EVALUATION.

| Training | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Accuracy | | | 1.00 | 12312 |
| Macro Average | 0.96 | 0.95 | 0.95 | 12312 |
| Weighted Average | 1.00 | 1.00 | 1.00 | 12312 |

### 8.4 Evaluation of the Result

The combined analysis shows three critical findings:

**1.** Overall Accuracy vs. Class-Level Balance: Although all ensemble models (RF, XGBoost, CatBoost) achieved ~100% accuracy, the real advancement lies in the balanced detection of minority attack types, validated through per-class precision/recall improvements.

**2.** Reduction of Misclassification Patterns: The confusion matrix analysis revealed that rare attacks were no longer absorbed into majority categories after SMOTE balancing. This correction eliminates blind spots that would otherwise persist in IoT intrusion detection systems.

**3.** Validation of Enhanced SMOTE: The superior performance on Classes 4 and 5, which initially had poor recall and F1-scores, validates the design of the customized SMOTE. Unlike standard SMOTE, the enhanced method generated synthetic samples with adjustable neighbor counts and interpolation factors, preventing overfitting while enriching the minority decision space.

**These results are the best; the reasons are clear below:**

- Significant feature reduction: The feature selection process using GWO reduced the dimensionality from 32 to 21 features, making the model both efficient and easier to interpret.
- Strong and stable model performance: All three ensemble models (RF, XGBoost, and CatBoost) maintained excellent stability and accuracy across all classes.
- Near-perfect accuracy for RF and XGBoost: Both models consistently reached ~100% accuracy, which is exceptionally rare in IoT IDS applications.
- Improved CatBoost performance: CatBoost showed notable gains in detecting minority attack classes, with marked increases in precision and recall, confirming the effectiveness of the applied modifications.
- Final model ranking: As illustrated in Table XII, the recommended order of deployment is RF first, XGBoost second, and CatBoost third.
- Practical deployment readiness: With only 21 features and almost perfect classification results, the proposed system offers a fast, effective, and ready-to-use model for IoT intrusion detection.

TABLE XII. THE FINAL RANKING OF MODELS BY PERFORMANCE AND QUALITY.

| Ranking | Model | Number of Features | Performance (Accuracy) | Notes |
|---|---|---|---|---|
| 🥇 **1** | **RF** | **21** | **1.00** ☑ | It was a perfect performance, the best overall |
| 🥈 **2** | **XGBoost** | **21** | **1.00** ☑ | It was a very excellent performance, like RF |
| 🥉 **3** | **CatBoost** | **21** | **1.00** ☑ | Clear improvement, very excellent performance |
| 4 | Decision Tree | 21 | 1.00 ☑ | Very good performance |
| 5 | SVM | 21 | 0.99 ☑ | Very good performance |
| 6 | Logistic Regression | 21 | 0.98 ⚠ | Relatively lower performance than other models |

### 8.6 Discussion

The integration of the Grey Wolf Optimizer for feature selection significantly enhanced model efficiency and reduced computational complexity by selecting only 21 optimal features from the original dataset. Combined with customized SMOTE oversampling and refined hyperparameter tuning, the detection accuracy of the RF, XGBoost, and CatBoost classifiers improved dramatically, indicating exceptional potential for real-world IoT cyberattack detection scenarios**.** Nevertheless, the achievement of 100% accuracy across multiple classifiers warrants careful interpretation. While feature reduction and improved data balancing mitigated some risk of overfitting, perfect accuracy may still reflect dataset-specific optimization rather than guaranteed generalization. In real-world IoT deployments, factors such as noisy data, evolving attack strategies, and resource limitations often reduce performance. Related high-performing IDS frameworks emphasize that even when benchmark results approach perfection, rigorous validation across diverse conditions is essential to ensure robust applicability.

## 8. CONCLUSION

This research presents a comprehensive and effective framework for cyber-attack detection in IoT networks, addressing key challenges such as high-dimensional data and class imbalance. By integrating GWO for optimal feature selection and an enhanced version of SMOTE, the proposed methodology significantly improves classification accuracy and generalization while reducing computational complexity. Experimental evaluation on the RT-IoT2022 dataset demonstrated that ensemble classifiers, particularly RF, XGBoost, and CatBoost, achieved near-perfect accuracy (100%) using only 21 optimally selected features. Although these results highlight the strong potential of metaheuristic-based feature selection and customized oversampling, achieving perfect accuracy also raises the possibility of overfitting to specific dataset characteristics. Therefore, caution is required when extending the findings to real-world IoT scenarios. Future research should prioritize testing under heterogeneous network conditions, cross-dataset validation, and deployment in resource-constrained environments to confirm scalability and resilience. Emphasis will also be placed on real-time adaptation and explainable AI to strengthen trust and transparency in mission-critical IoT security systems.

## CONFLICTS OF INTEREST

The authors specifically claim they do not have any conflicts of interest.

## AUTHOR CONTRIBUTIONS

Conceptualization Sura A. Sarab, Mustafa S. Ibrahim; methodology Nada H. Mohammed, Mustafa S. Ibrahim; software Sura A. Sarab, Mustafa S. Ibrahim; analysis Nada H. Mohammed, and, writing original draft Sura A. Sarab, Mustafa S. Ibrahim; validation and reviewing the manuscript by all authors, who then approved the final version of the manuscript.

## ACKNOWLEDGMENTS

The authors express their gratitude to Baghdad University for collaborating with them through their website at (http://uobaghdad.edu.iq).

### Funding

None

## REFERENCES

[1] G. S. Nadella and H. Gonaygunta, “Enhancing Cybersecurity with Artificial Intelligence: Predictive Techniques and Challenges in the Age of IoT,” *International Journal of Science and Engineering Applications*, vol. 13, no. 04, pp. 30–33, 2024.

[2] J. Deogirikar and A. Vidhate, “Security attacks in IoT: A survey,” in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, 2017, pp. 32–37.

[3] A. A. Nafea, M. M. Hamdi, B. saad Abdulhakeem, A. T. Shakir, M. S. I. Alsumaidaie, and A. M. Shaban, “Detection Systems for Distributed Denial-of-Service (DDoS) Attack Based on Time Series: A Review,” in *2024 21st International Multi-Conference on Systems, Signals & Devices (SSD)*, IEEE, 2024, pp. 43–48.

[4] A. H. Salem, S. M. Azzam, O. E. Emam, and A. A. Abohany, “Advancing cybersecurity: A comprehensive review of AI-driven detection techniques,” *Journal of Big Data*, vol. 11, no. 1, p. 105, 2024.

[5] Yaseen, M. G., Ali, A. H., & Alajanbi, M., “An Enhanced Hybrid Genetic-JAYA Algorithm for Feature Selection and SVM Parameter Optimization in Intrusion Detection Systems: Evaluation on the CICIDS Dataset”. *SHIFRA*, *2025*, 98-109. https://doi.org/10.70470/SHIFRA/2025/006.

[6] U. Inayat, M. F. Zia, S. Mahmood, H. M. Khalid, and M. Benbouzid, “Learning-Based Methods for Cyber Attack Detection in IoT Systems: A Survey on Methods, Analysis, and Future Prospects,” 2022. doi: 10.3390/electronics11091502.

[7] Li, Y., Lin, H., Liu, Y., & Zhang, X. (2024). *"A Lightweight Federated Learning Framework for IoT Intrusion Detection Using Attention-Based Neural Networks." IEEE Internet of Things Journal, 11(3), 4512–4525. DOI:* https://doi.org/10.1109/JIOT.2024.1001932.

[8] Zhang, L., Wang, R., & Elhoseny, M. (2024). *Hybrid attention-based LSTM-CNN model for detecting cyber threats in IoT networks.* IEEE Transactions on Network and Service Management, 21(1), 75–88. DOI: https://doi.org/10.1109/TNSM.2024.1001743.

[9] A. Gueriani, H. Kheddar, and A. C. MAZARI, *Enhancing IoT Security with CNN and LSTM-Based Intrusion Detection Systems*. 2024. doi: 10.48550/arXiv.2405.18624.

[10]T. Subalaxmi, R. Akalya, R. Kaviya, V. Santhosh, and P. Ramalingam, *Enhancing Cyber-Attack Detection in IoT Networks through Deep Reinforcement Learning*. 2024. doi: 10.1109/ICKECS61492.2024.10616565.

[11] A. Hammad, M. Falih, S. Abd, and R. Ahmed, "Detecting Cyber Threats in IoT Networks: A Machine Learning Approach," *International Journal of Computing and Digital Systems*, vol. 17, pp. 1–25, Dec. 2024, doi: 10.12785/ijcds/1571020041.

[12] Thaker Nay, "Enhancing IoT Security with AI-Driven Hybrid Machine Learning and Neural Network-Based Intrusion Detection System". Babylonian Journal of Artificial Intelligence, Vol. 2024, 2024, pp 158-167. DOI: 10.58496/BJAI/2024/017.

[13] Almalki, A., Kim, T., & Shaaban, E. (2023). *A distributed, lightweight deep learning approach for intrusion detection in resource-constrained IoT environments.* Computers & Security, 131, 103191. DOI: https://doi.org/10.1016/j.cose.2023.103191.

[14] Y. Kongsorot, P. Musikawan, P. Aimtongkham, I. You, A. Benslimane, and C. So-In, "An Intrusion Detection and Identification System for Internet of Things Networks Using a Hybrid Ensemble Deep Learning Framework," *IEEE Transactions on Sustainable Computing*, vol. PP, Aug. 2023, doi: 10.1109/TSUSC.2023.3303422.

[15] A. G. M. Mengara, Y. Yoo, and V. C. M. Leung, "IoTSecUT: Uncertainty-Based Hybrid Deep Learning Approach for Superior IoT Security Amidst Evolving Cyber Threats," *IEEE Internet Things J*, 2024.

[16] M. Chemmakha, O. Habibi, and M. Lazaar, "A novel hybrid architecture of conditional tabular generative adversarial network and 1d convolution neural network for enhanced attack detection in IoT systems," in *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*, IEEE, 2023, pp. 156–161.

[17] B. S. Sharmila and R. Nagapadma, "Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset," *Cybersecurity*, vol. 6, pp. 1–15, 2023.

[18] Ž. Vujović, "Classification model evaluation metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021.

[19] M. S. I. Alsumaidaie, K. M. A. Alheeti, and A. K. Al-Aloosy, "Intelligent Detection System for a Distributed Denial-of-Service (DDoS) Attack Based on Time Series," in *2023 15th International Conference on Developments in eSystems Engineering (DeSE)*, IEEE, 2023, pp. 445–450.

[20] G. Naidu, T. Zuva, and E. M. Sibanda, "A review of evaluation metrics in machine learning algorithms," in *Computer Science Online Conference*, Springer, 2023, pp. 15–25.

[21] M. Muntean and F.-D. Militaru, "Metrics for evaluating classification algorithms," in *Education, Research and Business Technologies: Proceedings of 21st International Conference on Informatics in Economy (IE 2022)*, Springer, 2023, pp. 307–317.

[22] M. Lutz, *Programming Python*. "O'Reilly Media", *Inc*., 2011.