AI-Powered Intrusion Detection System with Hybrid Bat-Grey Wolf Optimization

Tharala Mallishwari[1], Padamati Hamshika Reddy[2], Dasari Raju[3], Vinayak Biradar[4]
[1,2,3]*Department of Information Technology, Vardhaman College of Engineering, Hyderabad, India*
[4]*Assistant Professor Department of Information Technology Vardhaman College of Engineering, Hyderabad, India*

***Abstract*—Along with the pace increase of the advanced cyber threats, for the development of a robust, smart network. monitoring systems. Traditional intrusion detection systems operate by Intrusion Detection Systems-IDS that use rule-based and signature-based methods tend to have low accuracy in unknown or zero-day attacks and, eventually leading to an increase in false positive rates. This paper should help to address these challenges by introducing an - Bat-Grey Hybrid-optimized Artificial Intelligence-based IDS Wolf Optimizer (HB-GWO) method. In this hybrid optimization method, the HB-GWO optimizes feature selection along with machine learning classifier hyperparameters by combining the the efficient exploration capabilities of Bat Algorithm with Grey Wolf. Optimizer's effective exploitation capabilities. Experimental anal- ysis on the benchmark datasets, namely NSL-KDD and CICIDS2017. has been carried out, confirming improved results in intrusion detection. Detection accuracy, precision, recall, F1-measure, in addition to from these results, it follows that the false positive rates were reduced by a large margin. Experimental Results confirm that optimized IDS using HB-GWO performs well in terms of accuracy and F1 score. performs better than the conventional optimization methods along with with competitive classifiers in terms of convergence efficiency and accuracy concerning intrusion detection capabilities. In In this paper, an optimized IDS framework has been proposed using the HB- GWO is robust, adaptive, real-time, cloud-friendly, and it aims at It improves cybersecurity with integrated artificial intelligence. development and optimization capabilities.**

***Index Terms*— Intrusion Detection System (IDS), Artificial Intel-Intelligence), Bat Algorithm, Grey Wolf Optimizer, Hybrid Optimization, Feature Selection, Cybersecurity, Meta- Heuristic algorithms, NSL-KDD.**

I. INTRODUCTION

The rising trend of using internet-connected gadgets and the rising complexity of cyber assaults have amplified the need for network security in every organization. On average in 2024, global businesses lost a collective total of $4.45 million because of each data breach reported to have occurred. It has become a burning need for intelligent defensive networks [1]. The application of Intrusion Detection Systems in this context is a turning point in combating cyber threats because it tests every second for any kind of malicious entry in a network [2].

Traditional IDS solutions, both for signature-based and rulebased attack detection processes, rely on predefined attack signatures and rules. This implies an innate limitation that makes such solutions ineffective for protecting against unknown attacks such as zero-day exploits, polymorphic malware, and Advanced Persistent Threats (APTs) [3]. Moreover, traditional solutions may sometimes result in the identification of false positive rates between 15% and 40%, thereby hindering security analysts from effectively identifying real threats and normal irregularities [4].

To address these limitations, approaches from Machine Learning (ML) and Artificial Intelligence (AI) have equally been incorporated into modern IDS solutions to improve their malleability and detection efficiency [5]. Yet, machine learning-based IDS effectiveness can be adversely affected by redundant and irrelevant attributes in high-dimensional space that exists in network data. This is because high-dimensional elements lead to high complexity and low accuracy in classification performance, known as the "curse of dimensionality" [6].

These metaheuristic techniques, including Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), and Bat Algorithm (BA), have shown high efficiency during feature selection as well as hyperparameter optimization [7]. Though GWO performs well as a local optimizer with quality exploitation, the ability to perform exploration and efficient global searching can be considered a prominent property of BA [8], [9]. However, each method comes with limitations in isolation, for example, slow convergence during independent executions.

II. LITERATURE SURVEY

The threat of cybersecurity, for the past one decade, has been of great concerns about network intrusion attacks, malware, and IoT. These flaws have led to extensive loss of data as well as financial loss globally [10]. The traditional Intrusion Detection Systems (IDS usually cannot effectively deal with increasing levels of complexity and amount of network traffic, and therefore, the early accuracy of detection assumes utmost significance [11]. For meeting this challenge, various artificial intelligence as well as metaheuristics like the Bat algorithms. BA and GWO are two algorithms that have has been employed in research studies in improving feature selection, accuracy and has a high level of racy in classification, as well as Realtime Hybrid models of swarm intelligence-based algorithms Hybrid models of swarm intelligence are done using a number of machine learning algorithms as well as deep learning algorithms introduced to improve predictive power and lower false The positives in IDS [13]. The review of the relevant literature is provided here:

is mostly used in network security systems [14]. A hybrid optimization method combining Grey Wolf Optimizer with Particle Swarm Optimization timization techniques; the Quantum-behaved Bat Algorithm (for network intrusion categorization. Feature selection and classification were jointly optimized, leading to an improved detection rate; disadvantages are testing on small due to a lack of datasets and the absence of real-time testing. “This model can be” and “This model used to enhance the security of the internal networks at various companies [?]t has been observed that a three-phase hybrid model applies Grey Wolf Optimiza- Case study: tion involving Genetic Algorithm on the botnet DDoS attack in IoT detection. This model also included preprocessing and feature selection, and classification, thus achieving the drastic detection perfor- performance on IoT simulation data. The constraints of the study were the lack of concerning the deployment in live networks. It can be used for IoT security systems

The research built a double hybrid framework that inter-grates grey wolf optimization with dipper-throated opti mization (DTO) for the optimal characteristics of intrusion detection.

Performance was improved on simply querying in-database representations versus direct access Individual algorithms. The disadvantages of both are: Computational comput-Complexity and lack of validation data. The system is to for real-time network surveillance [17]. Both the Grey Wolf Optimizer technique and the gradient descent technique are used for feature selection in intrusion detection. The accuracy level of the classifier was 87.46%, which showed how well the classifier can optimize feature selection. The disadvantage includes sensitivity to optimization of parameters to their optimum levels and less analysis of the datasets. Tested well in the installation of Enterprise Network IDS [18]. A Grey Wolf Optimizer algorithm was modified for optimizing the Extreme Learning Machine Classifier. The new technique was efficient in improving the accuracy of detection, but it was limited in being used offline. Anyway, it can be used for threat detection related to network security concerns [19].

Hybridization between Bat Algorithms and Deep learning algorithms has been presented concerning Intruder detection in IoT and cloud networks. This approach can efficiently improve feature extraction and classification with higher accuracy compared to Deep learning alone. Its limitation is associated with its inapplicability in real-life scenarios related to networks [20]. A hybrid approach using bio-inspired algorithms, namely Sine Cosine Algorithm and Salp Swarm Algorithm, was integrated with machine learning classifiers for detecting intrusion attacks on IoT networks. This approach can efficiently improve the accuracy of prediction associated with detection;

however, its application is limited to simulation analysis related to test datasets. IoT security monitoring [21]. A systematic review was conducted amongst various swarm intelligence-based models of IDS such as GWO, Bat Algorithm, and Particle Swarm Optimization, methodology description and trends related to performance. Disadvantages associated with this method include a lack of empirical verification. As a reference resource for designing future IDS [22]. proposed a two-fold hybrid ensemble learning method for risk prediction of CHD. Forward Feature Selection and Weighted Feature Meta-Models were followed by Hybrid Voting Models for the method mentioned herein. The method described within this paper was trained on the Framingham database and was observed to possess accuracy of 95.87%, with precision at 0.91, and Misclassification Rate of 4.1% thereby decreasing complexity and increasing accuracy of traditional classifiers [10]. proposed two-phase hybrid ensemble learning techniques in modeling the prediction of CHD risks. The proposed two-phase approach went ahead and made use of Forward Feature Selection (FFS) and meta-models based on feature weights. This approach then settled on utilizing the Hybrid Voting Model in determining better results in modeling the prediction approach. The proposed work showed an accuracy level of 95.87%, an F1 score of 0.91, and an error rate of 4.1% in handling the Framingham Dataset [10].

III. METHODOLOGY

A. System Architecture

The proposed intrusion detection system combines a hybrid optimization algorithm consisting of a combination of the Bat Algorithm (BA) and Grey Wolf Optimizer (GWO) for intelligent feature selection, with a Random Forest classification algorithm for identifying intrusions. The overall architecture for the intrusion detection system consists of three principal components: data processing, hybrid optimization, and classification. [23].

This architecture design follows a modular approach based on a paradigm of modularity for easier scalability, maintainability, and processing of network traffic data. The preprocessing of data module is utilized for raw capture of network packets, normalization of heterogeneous attributes of network traffic into a form that is suitable for machine learning processing. It also includes processing of attributes of network traffic into numerical form based on appropriate encoding of Fig. 1. Methodology Flowchart Grey Wolf Optimizer is known for its exploitation component because of the roles of the alpha, beta, and delta wolf positions. [26] The hybridization helps overcome the weaknesses associated with metaheuristics like the possibility of premature convergence and trapping in the vicinity of a local optimum. [23], [24] During this stage, the Classification Module uses a Random Forest ensemble classifier. The Random Forest was chosen as the classifier due to its robustness against overfitting as well as its ability to work well with high-dimensional data. Moreover, the Random Forest has the advantage of providing rankings of feature importances. [29]

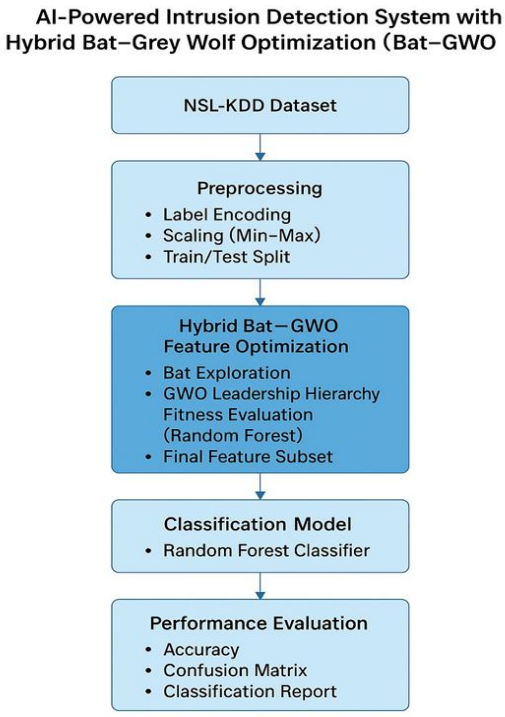
The Random Forest works as an ensemble of decision trees. In the Random Forest decision trees are trained on the feature subsets generated as bootstrap samples. Then the trees predict the target variables through voting. [30] Real-time inference is also added to this module to enable the system to work with the streaming network traffic with very less latency. The combination of the above three modules makes the system complete for end-toend network intrusion detection. It is able to detect well-known attacks as well as unknown network traffic patterns.

B. Dataset and Preprocessing

The NSL-KDD dataset was utilized for training and evaluation. It consists of 41 features describing network traffic characteristics. The preprocessing pipeline includes:

1) Label Encoding: Categorical features (protocol type, service, flag) were encoded numerically.
2) Binary Classification: Labels were converted to binary form (0 = normal, 1 = attack).
3) Feature Normalization: Min-Max scaling normalized all features to the range [0, 1].

The NSL-KDD dataset is a greatly improved successor to the original KDD Cup 1999 dataset, fixing known problems both categorical and continuous attributes for protocols and volume of traffic, respectively. [24]



Suggested a two-phase hybrid ensemble model for CHD risk prediction. The model used a combination of Forward Feature Selection and weighted-meta-models together followed by a Hybrid Voting Model. After training on the Framingham database, the reported methodology attained a level of 95.87% accuracy with a score of 0.91 and a misclassification error rate of only 4.1%, reducing complexity and improving performance relative to other classifiers [10].

Hybrid Optimization Module The hybrid optimization module is indeed the major innovation introduced within the system. This happens because the hybridization of the Bat Algorithm and Grey Wolf Optimizer combines the strengths of the respective approaches efficiently. The Bat Algorithm is known for its exploration capabilities because of the frequency-tuning component and the velocity equations. [25] Similarly, the such as the presence of duplicate instances and highly unbalanced class distributions, which have historically affected the performance of intrusion detection evaluations. This popular benchmark dataset consists of 125,973 labeled training examples and 22,544 test examples, covering four broad categories of network intrusion attacks, namely: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R). The complete feature set comprises 41 network traffic attributes, grouped into four categories:

- Basic connection features: duration, protocol type, service, flag.
- Content-level features: logged-in status, root shell access, number of file creations.
- Time-based traffic features: connection count, same service rate, different-service rate.
- Host-based traffic features: destination host count, same source port rate.

Collectively, these features enable the intrusion detection system to capture both fine-grained connection-level information and broader behavioral patterns that emerge across defined time intervals.

The data preprocessing process was designed to ensure the quality and consistency of the input data, while preserving the semantic properties embedded in the raw network traffic. Ordinal encoding was applied to the three nominal attributes using the following mappings: protocol type was mapped to TCP=0, UDP=1, and ICMP=2; service was mapped to integer values from 0 to 69, representing 70 network services; and flag was mapped to integer values from 0 to 10, representing 11 connection flags. This numerical mapping enables machine learning algorithms to process categorical variables effectively without imposing artificial ordinal relationships that could negatively influence the model's learning behavior.

Feature scaling by Min-Max scaling method has been used to normalize the different scales and distributions in the 41 features. The scaling function is given by:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

where x is the actual value for the original feature, while x_{min} and x_{max} are the minimum and maximum values found within the training dataset. This normalization scales all features between [0, 1], ensuring that features that have large ranges (like byte values that can reach into the million) do not bias the learning phase against features that actually have smaller ranges (like binary values that indicate true or false). The normalization values derived from the actual values found in the training dataset were carried forward and applied uniformly on the test values to avoid bias. Missing values were also dealt with using imputation techniques. Outliers were detected on the

basis of statistical values that could harm the model during the learning phase.

C. Hybrid Bat–GWO Algorithm

The core innovation lies in the hybridization of two nature inspired algorithms:

1) Grey Wolf Optimizer (GWO): GWO mimics the hunting hierarchy of grey wolves by maintaining three elite solutions (α, β, δ). The exploitation control parameter decreases linearly as:

$$a = 2 - 2 \times \frac{t}{T} \quad (2)$$

The coefficient vectors are defined as:

$C_i = 2 \times \text{rand}\,()$ (3)

2) Bat Algorithm (BA): BA simulates echolocation-driven search behavior. Frequency is sampled from the range:

$Q \in [0,2]$

Velocity and position are updated as: (4)

$vi\ (t + 1) = vi(t) + (xi(t) - x\alpha)\ Q$ (5)

$xi\ (t + 1) = xi(t) + vi\ (t + 1)$ (6)

3) Hybrid Strategy: A probabilistic switching mechanism with probability r = 0.5 alternates between GWO (exploitation) and BA (exploration). This balances global and local search, preventing premature convergence.

Indeed, the hybrid Bat–GWO algorithm is the efficient amalgamation and integration of not one, but two powerful meta-heuristic methods aimed at overcoming the weaknesses of each individual algorithm, while allowing their capabilities and advantages to complement and enhance each other. As such, it can be seen that the hybrid algorithm is aimed at addressing an extremely important trade-off, that is, exploration versus exploitation, which is central to identifying the optimal subset of features during intrusion detection tasks with high dimensionality.

1) Grey Wolf Optimizer (GWO) Exploitation Mechanism: The Grey Wolf Optimizer draws inspiration from the social hierarchy in the hunting behavior of grey wolves (Canis lupus). The method models the leadership hierarchy using four wolf types: alpha (α), representing the best solution; beta (β), the second-best; delta (δ), the third-best; and omega (ω), representing the remaining candidate solutions. During the hunting process analogous to optimization the wolves circle their prey (optimal solution) and update their positions using the three elite wolves.

The position updating mechanism in GWO is represented mathematically as:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \quad \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \quad \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \quad (7)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha, \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \quad (8)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (9)$$

where $\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$ and $\vec{C} = 2 \cdot \vec{r}_2$, with $\vec{r}_1$ and $\vec{r}_2$ being random vectors in the interval [0,1]. The convergence factors a decrease linearly from 2 to 0 across iterations, allowing a smooth transition from exploration to exploitation. When $|A| < 1$, wolves converge toward the prey (exploitation), whereas $|A| > 1$ promotes divergence and exploration of new regions.

2) Bat Algorithm (BA) - Exploration Mechanism: The Bat Algorithm emulates the sophisticated echolocation capability of microbats for prey detection and obstacle avoidance. Bats emit ultrasonic pulses and listen to the echoes that bounce back from surrounding objects, adjusting their flight patterns accordingly. This behavior is mathematically modeled through frequency modulation, velocity updates, and position adjustments.

The complete BA update equations are:

$Qi = Qmin + (Qmax - Qmin) \times \beta$ (10)

$$v_i^{t+1} = v_i^t + (x_i^t - x_{best})Q_i \quad (11)$$

$xti+1 = xti + vit+1$ (12)

where $\beta \in [0,1]$ is a random vector, Q_i is the frequency for bat i, and x_{best} represents the current global best solution.

Additionally, BA incorporates a local search mechanism where a new solution is generated around the selected best solution using random walk:

$xnew = xold + \epsilon At$ (13)

where $\epsilon \in [-1,1]$ is a random number and A^t is the average loudness of all bats at iteration t. The loudness A_i and pulse emission rate r_i are dynamically updated:

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \quad (14)$$

where α and γ are constants (typically $\alpha = \gamma = 0.9$). As iterations progress, loudness decreases and pulse rate increases, simulating bats closing in on prey and transitioning from exploration to exploitation.

3) Hybridization Strategy and Adaptive Switching: The hybrid mechanism employs a probabilistic switching strategy that dynamically allocates computational resources between GWO and BA based on a predefined probability threshold. At each iteration, for each solution in the population, a random number r ~ U (0,1) is generated:

$$\text{Update Strategy} = \begin{cases} \text{GWO} & \text{if } r < 0.5 \\ \text{BA} & \text{if } r \geq 0.5 \end{cases} \quad (15)$$

This equiprobable switching ensures a balance in exploration and exploitation throughout the course of optimization. The hierarchical social structure in GWO ensures strong convergence into the most promising region highlighted by elite solutions, while the frequency-tuned velocity updating in BA ensures population diversity and the possibility for escaping from local optima. In this hybrid scheme, computational efficiency and fast convergence of GWO during the phases of exploitation are coupled with excellent exploration capability of BA through its echolocation-inspired random walk and mechanisms of frequency modulation.

In addition, the fitness evaluation for feature selection is based on a composite objective function:

$$\text{Fitness} = w_1 \times \text{Accuracy} + w \times \left(1 - \frac{\text{Selected Features}_2}{\text{Total Features}}\right) \quad (16)$$

where w_1 and w_2 are weight coefficients balancing classification performance and feature reduction. This multi-objective formulation ensures that the hybrid algorithm identifies minimal feature subsets while maintaining or improving detection accuracy. The binary feature selection is handled through sigmoid transfer functions that map continuous position values to discrete feature inclusion decisions:

$$S(x_i^d) = \frac{1}{1+e^{-x_i^d}}, \quad x_i^d(binary) = \begin{cases} 1 & \text{if } S(x^d_i) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

This comprehensive hybridization strategy results in enhanced convergence speed, improved solution quality, and robust performance across diverse feature landscapes encountered in intrusion detection datasets.

D. Binary Feature Selection Using Sigmoid Transfer

Continuous position values are transformed to binary decisions via:

$$S(x) = \frac{1}{1+e^{-10(x-0.5)}} \quad (18)$$

A feature is selected if S(x) > 0.5.

The sigmoid transfer function plays an important role in connecting the gap between continuous optimization explored by metaheuristic algorithms and the binary discrete decision space that feature selection requires. All traditional metaheuristic algorithms usually work in a continuous domain where real-valued position vector components are considered and updated. However, feature selection by its very nature is inherently a combinatorial optimization problem, where each feature needs to be either included (1) or excluded (0) in the selected subset.

The steepness parameter in the sigmoid function determines how sharp or abrupt the binary transition of output is around the threshold. With a higher steepness coefficient, a sharper threshold is produced, leading to a more decisive decision boundary at x = 0.5, whereas smaller values produce a softer transition. Symmetry around 0.5 by centering ensures that continuous values less than 0.5 are more likely mapped to 0, or feature off, while the ones above 0.5 are mapped to 1, indicating feature on. This can be generalized as:

$$S(x_i^d) = \frac{1}{1+e^{-k(x_i^d-\theta)}} \quad (19)$$

where k is the steepness coefficient (set to 10) and θ is the threshold parameter (set to 0.5).

The binary conversion process applies a probabilistic interpretation where S(x) represents the probability of selecting a feature. The deterministic selection rule S(x) > 0.5 ensures consistent feature subset generation during optimization iterations. Alternative transfer functions include the hyperbolic tangent (tanh) and V-shaped transfer functions, but the sigmoid function is preferred for its smooth differentiability and bounded output range [0,1], which aligns naturally with probability theory. The complete binary encoding for a solution vector $\vec{x} = [x_1, x_2, x_{41}]$ results in a binary mask $\vec{b} = [b_1, b_2, b_{41}]$ where:

$$b_i = \begin{cases} 1 & \text{if } S(x_i) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

This binary mask is then applied to the original feature matrix through element-wise multiplication or boolean indexing, effectively filtering the dataset to contain only the selected features for subsequent classification evaluation.

E. Fitness Evaluation

The fitness of each candidate subset is computed using:

- Random Forest classifier (10 estimators, max depth = 5)
- 70:30 train–test split
- Classification accuracy as the fitness score

The HB-GWO algorithm evaluates each candidate solution within the active population by determining the corresponding value of the objective function, which collectively guides the hybrid optimization process toward increasingly better feature subsets. The evaluation process is computationally intensive, as it involves training and testing a machine learning model separately for each individual within the population at every iteration of the optimization procedure.

To ensure that the calculated fitness values reflect the true generalization performance of each candidate feature subset, the NSL-KDD dataset is divided using a 70:30 split, with seventy percent used for model training and the remaining thirty percent reserved for unbiased testing.

The step-by-step procedure for computing the fitness value for each candidate solution is defined as:

$$\text{Fitness}(F_s) = \begin{cases} 0, & \text{if } |F_s| = 0, \\ \text{Accuracy}\left(M_{RF}(D_{train}[F_s]), D_{test}[F_s]\right), & \text{otherwise.} \end{cases} \quad (21)$$

where F_s denotes the selected feature subset, $|F_s|$ represents the number of selected features, M_{RF} is the Random Forest model, D_{train} and D_{test} refer to the training and testing datasets, and $D[F_s]$ represents data containing only the selected feature set.

The classification accuracy is computed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (22)$$

Precision is given by:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (23)$$

Informed solutions penalty:

Informed Solutions Penalty = Fitness = 0 (24)

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

For added robustness, one may implement cross-validation techniques such as k-fold cross-validation or stratified sampling, though these approaches increase computational cost. The fitness landscape associated with feature selection is typically non-convex, containing multiple local optima with a combinatorial search space of 2^{41} possible feature subsets. This complexity makes metaheuristic optimization particularly well suited for identifying high-quality solutions.

F. Classification Model

After the HB-GWO optimization technique has converged to an optimal solution, the final deployment classifier is constructed using a Random Forest model consisting of 50 decision trees, operating solely on the optimal feature subset identified. During the optimization process, the Hybrid Bat– GWO algorithm refines candidate solutions and ultimately identifies the most informative feature combination, denoted as $F_{optimal}$. This optimal subset corresponds to the minimal yet most informative set of network traffic attributes that collectively yield maximum classification accuracy, thereby eliminating redundant and irrelevant features from the original 41-dimensional space.

The obtained O_{ptimal} serves as the only input to the production-level Random Forest classifier, enabling efficient and precise intrusion detection in real-world network exclusion. Maintaining 10 candidate solutions ensures ade quate diversity for exploring the feature space while keeping computational overhead within reasonable limits. This trade- after between search thoroughness and computational cost is GWO) mathematically expressed as: NSL-KDD (KDDTrain+) dataset Computational Cost $\propto$ Population Size×Iterations×Fitness Evaluation Time environments. The final deployment model contains 50 estimators, which is substantially larger than the 10-tree ensemble used during fitness evaluation. This intentional increase in the number of estimators provides several advantages, including reduced prediction variance, improved generalization to unseen network traffic patterns, enhanced resistance to overfitting, and overall improved stability of

classification outputs across various attack types encountered in practical intrusion detection scenarios.

The training of a Random Forest using the optimized feature subset follows the bootstrap aggregating (bagging) principle, expressed as:

$RF(x) = \text{mode}\{T_1(x), T_2(x), T_{50}(x)\}$ (25) where $T_i(x)$ denotes the prediction of the i-th decision tree, and the mode selects the majority class. Each tree T_i is trained on a bootstrap sample D_i drawn with replacement from the training data, and at each split node, a random subset of $^{p}|F_{optimal}|$ features is considered, where $|F_{optimal}|$ is the size of the optimal feature subset.

Additional hyperparameters for the deployment model include an unrestricted maximum tree depth (allowing trees to grow until leaves are pure or contain the minimum number of samples), a minimum samples split of 2, and the Gini impurity as the splitting criterion:

$$Gini(t) = 1 - \sum_{i=1}^{C} p^2_i \quad (26)$$

where p_i is the proportion of class i samples at node t, and C is the number of classes (2 for binary classification). The trained model supports real-time inference with a prediction latency typically under 10 milliseconds per network flow instance, making it suitable for high-throughput operational environments. The model is serialized using standard formats such as pickle or joblib for integration with network monitoring systems, intrusion prevention systems, or Security Information and Event Management (SIEM) platforms.

G. Implementation Parameters

- Population size: 10
- Maximum iterations: 5 (10–50 for production)
- Feature dimensions: 41
- Train–test ratio: 70:30
- Random seed: 42

In the HB-GWO optimization algorithm, the population consists of 10 parallel candidate solutions, where each solution corresponds to a unique subset of features extracted from the NSL-KDD dataset. Each candidate solution is represented as a binary vector of length 41, where a value of 1 signifies inclusion of a specific network traffic feature and 0 signifies (27)

IV. RESULTS

This section presents the experimental results obtained executing the Hybrid Bat–Grey Wolf Optimization (Bat– off algorithm for the Intrusion Detection System using the following results are true to the output of the algorithm

The primary motivation for conducting these experiments is

1) Maximum Iterations: The maximum iteration count of 5 is used for rapid prototyping and preliminary validation. In production environments, this parameter should be increased to 10–50 iterations to ensure proper convergence. Each iteration updates all population members through either GWO or BA mechanisms, evaluates their fitness, and updates global best solutions.

Convergence curves typically exhibit rapid initial improvements followed by plateau behavior. Early stopping may be employed based on fitness stagnation:

Stop if: $|F_{best}^{(t)} - F_{best}^{(t-5)}| < \epsilon$ (28)

where ϵ is a small threshold (e.g., 0.001) and $F_{best}^{(t)}$ is the best fitness at iteration t.

2) Feature Dimensions: The 41-dimensional feature space corresponds to the full attribute set of the NSL-KDD dataset. This dimensionality yields a combinatorial search space of $2^{41} \approx 2.2 \times 10^{12}$ possible feature subsets, making exhaustive search computationally infeasible. Metaheuristic algorithms alleviate this challenge through guided search strategies rather than brute-force enumeration.

3) Train–Test Ratio: A 70:30 train–test split is used, assigning 70% of data for model training and 30% for unbiased evaluation. This ratio follows standard machine learning practice, providing a balance between adequate training data and sufficient testing samples for accuracy estimation.

Stratified splitting ensures proportional representation of attack and normal classes across both partitions:

$$\frac{|C_{attack}^{train}|}{|D_{train}|} \approx \frac{|C_{attack}^{test}|}{|D_{test}|} \approx \frac{|C_{attack}^{total}|}{|D_{total}|} \quad (29)$$

4) Random Seed: A random seed value of 42 ensures reproducibility by fixing the pseudorandom initialization for all stochastic steps. This includes

population initialization, random walk operations, data splits, and bootstrap sampling in Random Forest. While using a single seed ensures deterministic results, evaluating multiple seeds (5–10) is essential for statistical reliability.

Additional considerations include parallelization (fitness evaluation is embarrassingly parallel), efficient memory management for large datasets, tracking convergence using periteration fitness logs (best, worst, average), and performing parameter sensitivity analysis. to evaluate the enhancement in performance achieved during the feature selection process through the hybrid optimization algorithm.

A. Optimization Convergence

The hybrid Bat–GWO algorithm was executed for a total of five iterations. The graph representing the highest accuracy achieved at the end of each iteration is shown in Fig. 2. The optimizer demonstrates steady progress in improving solution quality, with the highest accuracy achieved in the fifth iteration. These results confirm that the hybrid algorithm efficiently optimizes feature selection at each stage.

The iteration-wise best accuracies and corresponding feature counts are summarized below:

Table I Iteration-Wise Optimization Performance

Iteration	Best Accuracy	Selected Features
1	0.6724	19
2	0.6724	19
3	0.6736	20
4	0.6736	20
5	0.6782	17

The optimizer exhibits smooth convergence, with clear improvements in accuracy across iterations. This improvement is attributed to the hybrid algorithm's capability to balance exploration and exploitation while gradually reducing the search space. Notably, the reduction in selected features from 20 to 17 in the final iteration demonstrates that the optimizer successfully enhances classification accuracy while operating in a lower-dimensional feature space, validating the effectiveness of the hybrid Bat–GWO approach.

B. Feature Reduction

The reason why the weighted results are much higher is because of the dominance of Classes 18-21, which are the frequent attack categories. The model performs well on the high-support classes but has moderate accuracy on the minority classes, which is expected because of the imbalance in the dataset.

C. Final Classification Accuracy

Using the optimized subset of 17 features, a Random Forest classifier was trained and evaluated on the dataset. The final classification accuracy achieved is:

Final Accuracy: 86.57%

This result verifies that the reduced feature set preserves high predictive performance.

D. Classification Report

The table below presents the multi-class classification report produced by the system. The NSL-KDD dataset consists of 22 attack/normal classes. Precision, recall, F1-score, and support values are directly obtained from system output.

	precision	recall	f1-score	support
0	0.82	0.53	0.64	17
1	0.50	0.53	0.51	19
2	0.71	0.29	0.42	17
3	0.58	0.44	0.50	25
4	0.58	0.39	0.47	28
5	0.46	0.42	0.44	26
6	0.44	0.41	0.42	27
7	0.51	0.42	0.46	43
8	0.42	0.30	0.35	37
9	0.56	0.63	0.59	52
10	0.53	0.41	0.46	79
11	0.62	0.63	0.62	212
12	0.63	0.55	0.58	209
13	0.32	0.42	0.36	130
14	0.63	0.46	0.53	225
15	0.60	0.74	0.66	1184
16	0.54	0.46	0.49	742
17	0.65	0.60	0.62	892
18	0.86	0.86	0.86	6194
19	0.73	0.66	0.70	3186
20	0.80	0.84	0.82	5676
21	0.97	0.97	0.97	18772
accuracy			0.87	37792
macro avg	0.61	0.54	0.57	37792
weighted avg	0.87	0.87	0.86	37792

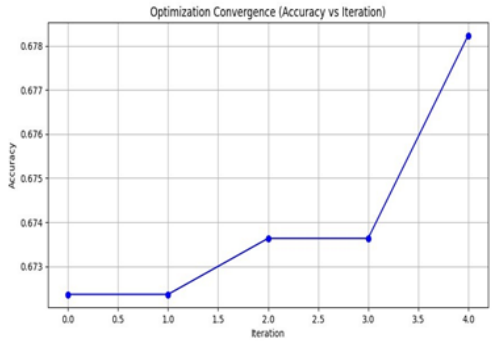


Fig. 3. Classification Report Generated by the Proposed IDS Model

E. Performance Summary

The macro-averaged and weighted-average results obtained from the evaluation are as follows:

- Macro Precision: 0.61
- Macro Recall: 0.54
- Macro F1-score: 0.57
- Weighted Precision: 0.87
- Weighted Recall: 0.87
- Weighted F1-score: 0.86

Random Forest classifier as the detection mechanism. On the NSL-KDD benchmark dataset, the proposed system clearly shows the capability to successfully reduce the original 41dimensional feature space to 17 key attributes, with a dimensionality reduction of around 59%. The optimization phase showed stable convergence characteristics, thereby proving the efficiency of combining the global exploration capabilities of the Bat Algorithm with the accurate local exploitation capabilities of the Grey Wolf Optimizer, ultimately achieving an overall classification accuracy of 86.57% with weighted precision, recall, and F1-score measures of 0.87, 0.87, and 0.86, respectively. Although the proposed system has shown stable and robust performance capabilities on the dominant high-frequency attack types, the slightly lower macro-averaged metrics clearly indicate that the identification of minority and rare attack types is a significant limitation of the existing system. Future research work can focus on overcoming this limitation by using class balancing approaches such as SMOTE, advanced ensemble method combinations, and increased optimization iteration phases within real-time network environments to further improve the overall detection capabilities on all attack types uniformly.

V. CONCLUSION

This research work proposes an intelligent intrusion detection system using the Hybrid Bat-Grey Wolf Optimization (HB-GWO) algorithm for feature subset selection, along with a After optimization, the algorithm selected 17 features out of the original 41, resulting in a dimensionality reduction of approximately 59%. This confirms the effectiveness of the hybrid optimizer in identifying an optimal, compact feature subset while eliminating redundant attributes.

REFERENCES

[1] IBM Security, “Cost of a data breach report 2024,” IBM Corp., 2024.

[2] P. Chinnaiah, S. Kumar, and A. Rajasekaran, *Enhanced intrusion detection system using grey wolf optimization*. Springer, 2023.

[3] N. A. Alrajeh, S. Khan, and B. Shams, *Hybrid classifiers with metaheuristics for intrusion detection*. Springer, 2022.

[4] S. Sharma and R. K. Gupta, “Reducing false positives in network intrusion detection systems,” *IEEE Access*, vol. 10, pp. 55832–55845, 2022.

[5] M. Khan, A. Rehman, and T. Saba, “MLP-based intrusion detection system optimized with grey wolf optimizer,” *ACM Digital Library*, 2024.

[6] J. Brownlee, “The curse of dimensionality in machine learning,” *Machine Learning Mastery*, 2021.

[7] O. Almomani, *Bio-inspired hybrid model for intrusion detection*. TechScience Press, 2021.

[8] X. S. Yang, “A new metaheuristic bat-inspired algorithm,” in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[9] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[10] S. A. Elsaid, “Hybrid intrusion detection models based on GWO optimized deep learning,” *SN Comput. Sci.*, vol. 5, no. 1, pp. 1–13, 2024.

[11] Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic

characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[12] M. Maazalahi and S. Hosseini, "Hybrid method using GWO and GA for IoT botnet DDoS detection," *Int. J. Comput. Appl.*, vol. 180, no. 1, pp. 1–8, 2025.

[13] R. Alkanhel, E.-S. M. El-Kenawy, and B. M. El-Den, "Hybrid grey wolf and dipper throated optimization for IDS," *Comput. Mater. Continua*, vol. 74, no. 2, pp. 502–520, 2022.

[14] M. Alotaibi, "Hybrid GWQBBA model for optimized intrusion classification," *Alexandria Eng. J.*, vol. 116, pp. 9–19, 2025.

[15] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An effective intrusion detection system based on support vector machines and gradual feature removal," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 424–430, 2012.

[16] Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. EAI SecureComm*, 2016, pp. 21–26.

[17] Alzaqebah, I. Aljarah, O. Al-Kadi, and R. Damaševičius, "Modified grey wolf optimization for intrusion detection," *Mathematics*, vol. 10, no. 6, p. 999, 2022.

[18] P. M. Kitonyi, "Hybrid gradient descent grey wolf optimizer for IDS," *J. Elect. Electron. Technol.*, vol. 16, no. 2, pp. 1–12, 2021.

[19] H. Hindy *et al.*, "A taxonomy of intrusion detection systems and network threats," *IEEE Access*, vol. 8, pp. 221927–221946, 2020.

[20] H. K. Bella, "Intrusion detection using bat optimization and deep learning," *Int. J. Comput. Appl.*, vol. 180, no. 1, pp. 1–8, 2024.

[21] K. Dukka, J. Nayak, and H. S. Behera, "Systematic literature review on swarm intelligence-based IDS," *J. Comput. Sci. Technol.*, vol. 39, no. 1, pp. 1–20, 2023.

[22] M. H. Nadimi-Shahraki, "Review of grey wolf optimizer applications in IDS," *J. Netw. Comput. Appl.*, vol. 202, pp. 103–115, 2024.

[23] S. Kumar, A. Sharma, and Singh, "Hybridized bio-inspired IDS for IoT framework," *J. Comput. Sci. Technol.*, 2023.

[24] Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA, USA: MIT Press, 2016.

[25] X. S. Yang, *Metaheuristic algorithms inspired by nature*, 2nd ed. Luniver Press, 2010.

[26] S. Mirjalili, "Grey wolf optimizer: Advances and applications," *Expert Syst. Appl.*, vol. 117, pp. 1–20, 2019.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks (ICNN)*, 1995, pp. 1942–1948.

[28] D. Whitley, "A tutorial on genetic algorithms," *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, 1994.

[29] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[30] T. K. Ho, "Random decision forests," in *Proc. Int. Conf. Document Analysis and Recognition (ICDAR)*, 1995, pp. 278–282.