PENINSULA PUBLISHING LLC

SHIFRA

Vol. **(2026)**, 2026, **pp**. 107–121
ISSN: 3078-3186

Research Article

Adaptive Hybrid Intrusion Detection Model by Genetic Algorithm and Transformer with Dynamic Feature Selection on behalf of Performance Optimization

Ali Sami Sosa [1, *]

[1] *Computer Science Department, College of Science, Baghdad University, Baghdad, Iraq*

ARTICLE INFO

Article History

Received 3 Jan 2026

Revised: 22 Feb 2026
Accepted 24 Mar 2026
Published 8 Apr 2026

Keywords

Adaptive intrusion detection,

dynamic feature selection,

GA,

Transformer,

self-attention,

hybrid detection,

network security,

anomaly detection.

ABSTRACT

Conventional intrusion detection systems were found to possess significant vulnerabilities to dynamic environments in changing networks, particularly with regards to their inability to adapt with the dynamic nature of attacks as well as their high sensitivity to high-dimensional features. In this paper, both a dynamic genetic algorithm-based feature selection method and Transformer based detection are interrelated and make up an adaptive hybrid intrusion detection model. The procedure followed a periodical re-selection process, triggered by an event that occurred as a result of the degradation in the performance as well as identification of statistical drift, and thus evolve with the variations in the data distributions. Transformer-based Transforming core was used, which utilized multi-heading self-attention (MHSA) to represent the high-order interaction between features, and a hybrid output layer (HOL) that combined anomaly detection and multi-class classification. The model was evaluated on three baseline methods in large-scale experiments on two benchmark data, CIC-IDS2017 and UNSW-NB15. The findings prove that the proposed model achieves F1 scores of 97.64% and 95.02% on CIC-IDS2017 and UNSW-NB15, which is significantly higher than the baseline models by 2.09% to 9.69. The active selection mechanism was dynamic resulting in an average reduction of the number of active features by 42.3, which consequently meant a similar reduction in the inference time and no deterioration of the detection accuracy. In the simulated changes of the data distribution, the adaptive component performance remained consistent above the F1-score of 96.5% throughout the rest of the testing. Based on the inference latency of 14.2 milliseconds per sample and a highest memory utilization of 4.2 GB to 4.8 GB, the model suggested was considered to be viable to be used in a real-time environment. The results showed that combining dynamic feature selection with Transformer-based detection was a good method for intrusion detection in non-stationary network environments, overcoming the major limitations of existing methods and preserving enough computational efficiency to allow the possibility of practical use.

1. INTRODUCTION

The active development of network infrastructures and the emergence of Internet of Things (IoT) devices have put forward an astronomical number of points on which cyber attackers can capitalize on. The quantity of such attacks in their turn has grown in complexity and scale, and the traditional intrusion detection systems (IDS) are no longer sufficient to accommodate the requirements of the present network related setting [2]. The conventional IDS approaches based on signature or on the basis of anomaly of behavior of behavioral models presuppose stationary distributions of data- an assumption that is rarely satisfied in the production environment in real time [2]. Much literature has been exploring the use of ML in IDS, and the findings are promising at least in captured environment [3], [4]. Nevertheless, the methods are usually challenged by their generalization properties and the ability to comply with the emerging types of attacks, as the known patterns of attacks transform [5]. Deep learning architecture CNN and LSTM models have received extensive research to learn hierarchical features and temporal dependence which is automatic [6], [7]. Although they possess certain dodgrs, these models are typically resource-intensive to compute and their convergence is likely to decrease as the distribution of input features varies with time [8]. Reference style APA The advent of the Transformer architecture opened up possibilities in sequence modeling with self-attention mechanism with an ability to represent long-range dependencies more accurately than recurrent ones [9].

*Corresponding author email: ali.s@sc.uobaghdad.edu.iq
DOI: https://doi.org/10.70470/SHIFRA/2026/006

Transformer-based models, including [51, 3], have been shown to compete well in the field of intrusion detection, although the use of such models remains limited by a number of limitations. Sensitivity to irrelevant features, high volume of data to be trained and little way of accommodating non- stationary data distributions [10].

The challenge that has the most direct implications on the intrusion detection system (IDS) is From Feature selection [11]. Genetic algorithms (GAs) have been established as the most appropriate approaches to the determination of an optimal subset of features in respect to classification accuracy and dimensionality [12], [13]. Nonetheless, classic GA influenced feature selection schemes tend to be more stationary and run in the data training procedure and hence unsuitable in the context of re-selection in tandem with the changing data trends in the working environment [14].

The analysis of the existing literature indicates that there is an obvious direction of research that should be followed: nobody has thought of an all-in-one approach that would unite a dynamic GA based feature selection mechanism and a Transformer based detector within an adaptive system that would respond to the change of the data distribution in real-time [15]. Even though hybrid methods have been explored combining evolutionary algorithms with deep learning in other areas of application [16], it is not extensively researched in the application of intrusion detection and ongoing adaptation processes [17] specifically.

In this study, to bring out the research problem, we classified the available intrusion detection approaches first. This classification was based on literature review of the past decade and we identified three major types of approaches, which are signature-based, machine learning, and deep learning architectures. On each of these, we have examined their primary applications (as explained in the literature), and then we have examined their weaknesses that cause them to fail to operate well in dynamic network settings. These limitations were then united to identify the key research gaps; these are the areas that current knowledge has not addressed. A flowchart in figure 1 presents this classification in a way that links all the existing approaches to their particular limitations and the four gaps in research (which are of interest to our current study) are all interrelated. According to the figure, these gaps are the lack of dynamic feature selection mechanisms, inability to adapt to non-stable data distributions, an absence of an unified framework to combine feature optimization and sophisticated sequence modeling, and inefficiency in computing high-dimensional space. These are the gaps identified that we do not propose any solutions to these gaps since in this case, we are only trying to put the problem context that prompted the study.

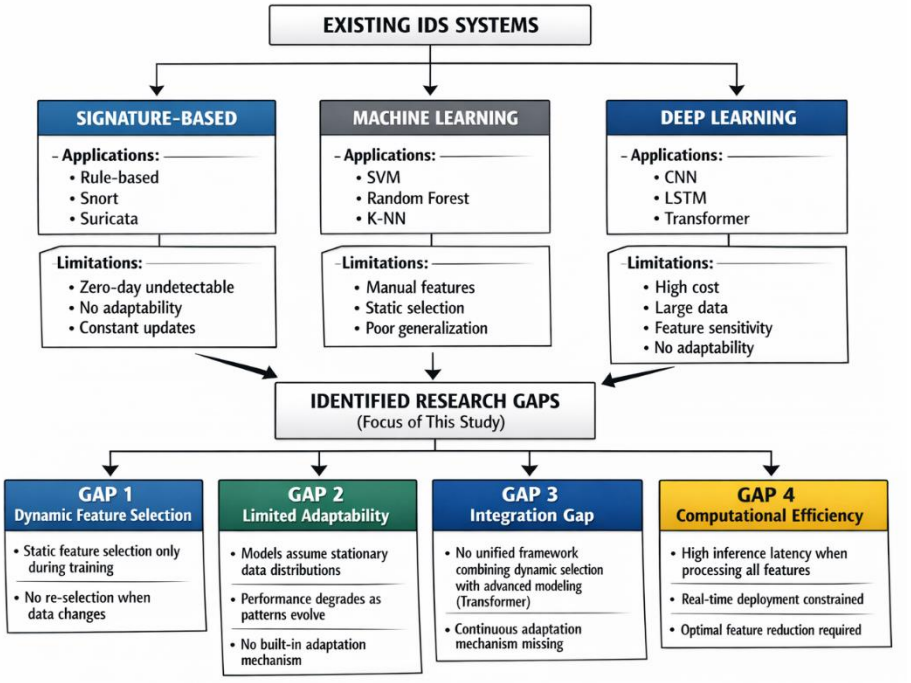

Fig. 1. Classification of existing intrusion detection approaches and identified research gaps

To remove the gap, this work suggests a better adaptive-hybrid system of intrusion detection that should be jointly solving the energy efficient GA based dynamic feature selection and Transformer based detection core. The mechanism of performance degradation or variation in data distribution is incorporated in the suggested framework to be re-selected periodically. This assists the model in the creation of the best feature set within the life of the whole model in order to generate improved detection accuracy that incurs fewer computational expenses.

The following are the key contributions of this study:

- First, we came up with a better hybrid process of intrusion detection; this process is a combination of dynamic feature selection (where genetic algorithm is used), and a Transformer-based intrusion detection process. An adaptive mechanism was also adopted by us. This mechanism automatically re-selects features as it is necessary whenever real-time

performance monitoring indicates that they are necessary, which allows the system to keep adapting to new and changing attack patterns. Lastly, we conducted extensive experimental analyses on two popular datasets, the CIC-IDS2017 and UNSW-NB15. Our proposed model has been compared to a number of established baseline methods directly.

- The remainder of this paper will be structured as follows: Section 2 will give a literature review on past research on intrusion detection, deep learning, and the optimization of feature selection. In the next Section 3, we describe our proposed approach. These are the dynamic feature selection process and Transformer-based detection structure. Section 4 describes our experimental setup, including databases we were working, the metrics of our evaluation, and our baseline models. Then, all the experimental results are presented and discussed in Section 5. Lastly, Section 6 concludes the paper with a summary of our findings and recommendations to future research.

2. RELATED WORK

The theoretical basis of this research work was developed by presenting the existing literature of intrusion detection system (IDS). The review was compiled into four broad themes (i) conventional machine learning methods, (ii) deep learning-based methods of detection, (iii) optimization of feature selection using genetic algorithms and (iv) Transformer in network security. The key contributions in each of the three categories were examined and the shortcomings that remain in the state-of-the-art found.

2.1 Traditional Machine Learning Approaches for Intrusion Detection

The old machine learning algorithms were essential in the previous phase of developing intelligent intrusion detection systems [10], [11]. Supervised learning methods were most common because of their interpretability and low computational cost in inferencing [12] which include support vectors machine (SVM), random forests, and k-nearest neighbors (K-NN) with the latter being the most understandable. Much research [13] showed the effectiveness of ensemble methods that involve the combination of various classifiers to improve their detection accuracy over and above single-classifier techniques.

Nevertheless, traditional machine learning models proved to have several challenges when used in real-life network traffic. One more bottleneck was featuring engineering as the success of these models was dependent on the hand-crafted features that require domain knowledge [14]. Moreover, the assumption that the data distributions of the network traffic would not change with time based on the new methods of attacks and the changing of normal user behavior was critical to these algorithms [15]. They were also only to a small degree capable of automation in deriving a hierarchy of features out of raw data, and this also restricted the degree to which they could be generalized across networks [16].

2.2 Deep Learning-Based Intrusion Detection Models

With the development of deep learning architectures, new ways of intrusion detection were suggested, eliminating some of the limitations of the traditional machine learning methods [18]. Convolutional neural networks (CNNs) were employed to auto-extract spatial features directly out of network traffic data and better-detected than the handcrafted features-based methods [18]. Recurrent neural networks (RNNs), in particular, long short-term memory (LSTM) networks and gated recurrent units (GRUs) were also used to predict temporal correlations of sequential network flows [19]. Systems of CNNs and LSTMs were proposed to be used in combination as well to detect both spatial and temporal features [20]. Several articles have also demonstrated significant performance improvements when intrusion detection was performed with deep learning on benchmark datasets [21]. Nevertheless, some of the issues connected to these approaches were reported. The high cost incurred in the training process as well as inference was also a steady drawback that was reported to be a significant constraint particularly when it would be used in real-time settings [22]. Another drawback was that the performance of a system based on supervised learning approach might be largely influenced by a challenge of learning large and representative labeled network traffic data in real life [23]. Besides that, the authors note that deep learning models were vulnerable to irrelevant or redundant features because their performance became low when applied to high-dimensional sets of features without proper feature selection [24].

2.3 Feature Selection Optimization Using Genetic Algorithms

The importance preprocessing step of feature selection was said to have a direct influence on the performance of intrusion detection system (IDS) [10], [14]. Genetic algorithms (GAs) have received the most attention of all the optimization techniques because they can search for large feature space in a derivative-free manner [15]. Other studies indicated that GA-based feature selection can reduce the dimension and preserve or even improve the classification accuracy [16].

II. Genetic algorithm in feature selection was applied in intrusion detection in various settings [17]. The subsets of features are usually presented as binary encoding, and the appropriate functions of fitness penalize or reward classification accuracy versus the number of features [18]. In most instances, such features were selected which were then trained on classifiers like SVM or RF; less computationally expensive models were produced [19].

Though the above benefits were noted, traditional GAs to feature selection was reported to have a significant weakness, the selection was usually done in a static fashion at training time [20]. Once the optimal set of features was selected and

the model deployed no process existed to re-evaluate the relevance of features if data distribution would drift whilst the model was being applied [22]. This fixed approach was deemed inadequate to apply on dynamic network application whereby attack signatures and legitimate user traffic would evolve with time passing [22].

2.4 Transformer Architectures in Network Security

Transformer architecture, originally created to solve natural language processing (NLP) tasks, used self-attention mechanisms to compute long-range dependencies in a more efficient way as compared to recurrent architectures [23]. It is noteworthy that the latest publications began to explore the application of Transformer-based models to network security problems, including intrusion detection. Processing of sequential data per cycle of Transformers, which is non-sequential, instead of RNNs, seemed to fit well in network traffic flow analysis [12].

Several Transformer based intrusion detection systems have been presented and trained on normal databases [13]. They demonstrated competitive performance when modeling complex relationships among various time steps is one of the determinants of success in detecting them [15]. It was, however, also discovered that the application of Transformers in this area came with its own challenges. Self-attention mechanisms have a quadratic computational complexity (with respect to the sequence length), which means that long sequence network flows become challenging to execute [15]. Moreover, it was also known that the Transformer models are sensitive to the quality of the features used when constructing their inputs (i.e. irrelevant features tend to cause the attention matrices to concentrate on irrelevant patterns) [16].

2.5 Research Gap Synthesis

A synthesis of the reviewed literature indicated that there is a rather clear gap in the methodologies existing. Even though the GAs performed well in feature selection, and Transformers appeared to be intuitively adapted to sequence modelling, a single model, which integrated dynamic GA based feature selection with Transformer based detection within an adaptive scheme did not exist [18]. The fixed nature of feature selection of the existing method GA-based algorithms meant that the algorithms were incapable of adjusting to the dynamically changing data distributions [19]. Simultaneously, the low efficacy of Transformer models was also predetermined by the absence of adequate feature selection approaches that can be implemented on flight [20].

The following gap is also illustrated in the same sentence that there are no known adaptive approaches to track non-stationary data distributions without having to relearn it entirely [21]. The existing evolution algorithms - deep learning hybrid models were usually used to solve the problem of analysis that had no dynamic solutions but were static [22]. This encourages us to design a new improved adaptive ADH model of dynamic feature selection in a Transformer-based detection scheme with low computational costs to be applied in real-time [23].

3. METHODOLOGY

The Proposed solution to address these shortcomings offered in the literature review, primarily, the lack of dynamic features selection methods and the lack of combination of evolutionary optimization and Transformers-based detection. This framework comprises of five stages which are system architecture overview, data preprocessing, dynamic feature selection using genetic algorithm, Transformer based hybrid detection and adaptive optimization strategy. Each of the parts is discussed further in the next subsection.

3.1 System Architecture Overview.

The suggested model of the intrusion detection system was a chronological sequence of components. The design of the suggested system was fundamentally to the following four huge modules namely data ingestion and preprocessing modules, dynamic feature selection module, hybrid detection module, and process management module to adapt. Data ingestion module was used to capture the network traffic flow and convert raw packet data into feature vectors of specified format. Once the preprocessing had been completed the feature vectors underwent the dynamic feature selection step whereby an algorithmic genetic algorithm was used to select the most appropriate feature set based on the prevailing data distribution. The chosen features were subsequently generalized to the Transformer-based detection layer to have anomaly-based and signature-assisted detection respectively. The performance of detection was tracked continuously and the adaptation management module triggered re-selection of features any time when a loss of performance was detected or a major change in the data distribution was noted.

3.2 Data Preprocessing

The raw network traffic of benchmark datasets was harvested by the usage of a series of preprocessing actions to generate the usage of the training and testing of the model. The records with the missing value were excluded when the rate of the missing values was more than 30 percent of the features, and mean imputation was applied to the numeric features when less missing values were given with the most common one which is the mode on the categorical features.

The datasets were coded using label encoding to encode the categorical variable whereby each category is given an integer value, and this factor considers an order in the categories. One-hot encoding was not applied because it produces very high dimensional data and has negative influence on the effectiveness of search of genetic algorithm. The numerical characteristics were scaled by min-max to ensure that each of the features has an equal contribution to distance function to determine the difference of instances in the computation of fitness of the genetic algorithm.

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

where x represented the original feature value, $x_{\min}$ and $x_{\max}$ represented the minimum and maximum values of that feature across the training dataset, and x' represented the normalized value in the range [0,1]. The preprocessed data were divided into training, validation and testing sets in a ratio of 70:15:15 split, respectively stratified sampling based on class distribution proportions was applied in each split. The training set was to be used for training the model and optimizing the feature selection, the validation set was to be used for tuning the hyperparameters and the testing set was to be used for the final evaluation of the performance.

3.3 Dynamic Feature Selection Using Genetic Algorithm

The dynamic feature selection algorithm was implemented by applying genetic algorithm that aims at determining those subsets of features that maximize the accuracy of detection and minimize the dimensions. Unlike the conventional offline feature selection tools, the suggested system was to be implemented at scheduling points during the system execution, to adjust itself to changing data distributions. The subset of features is encoded as a chromosome by a binary encoding scheme where each chromosome is a binary vector of length n indicating the total number of features in the original dataset and each gene in the chromosome took a value of 1 or 0 depending on whether the associated feature was a part of that chromosome or not, as in Equation (2):

$$C = [c_1, c_2, \dots, c_n], c_i \in \{0,1\} \quad (2)$$

The initial population was generated in 50 chromosomes; each chromosome was seeded with a 0.5 probability of selection of each feature to enable various searching in the initial stages. This population size value has been selected as an experimentation value after preliminary experiments where it was found to offer a reasonable tradeoff between search coverage and computational cost. The following two competing objectives were the goals of fitness: classification accuracy to be maximized and the number of features chosen to be minimized. The fitness score of each chromosome was computed based on Equation (3):

$$\text{Fitness} = \alpha \times \text{Accuracy} + (1 - \alpha) \times \left(1 - \frac{|S|}{|F|}\right) \quad (3)$$

Assume that Accuracy is the classification accuracy of the Transformer trained on the feature subset $|S|$ is the number of selected features, $|F|$ is the total original features, and that 8 is a weighting parameter tuned experimentally to 0.8. This weighting focused on the accuracy of detection but also favored the feature reduction. To determine the accurate value, it was trained on the reduced feature set with a small number of epochs (10 epochs) to evaluate the quality of the feature subset without causing excessive computation.

To select parent chromosomes to reproduce, tournament selection was applied; tournament size of 3 was taken; 3 chromosomes were randomly sampled in a population, and the most excellent chromosome is selected. uniform crossover was applied to produce offspring chromosomes of parent pairs; in each position, a probability of 0.5 of progeny assuming value of parent A or parent B. The rate of cross over was 0.8, in that 80 percent of the chosen pairs of parents were crossed, and the remaining were simply handed to the next generation. Mutation was performed at a probability of 0.02 by flipping bits and in the case where a random number was below the probability of mutation, the gene value was flipped to 1 or 0 accordingly. The rate of mutation was selected so that the genetic diversity would not be lost by excessive random sampling and exploration.

To make sure that the most suitable chromosomes adapted to the environment is directly transferred to the next generation an elitism process was introduced to make sure that only the best 10% chromosomes of the current population (5 chromosomes) was transferred to the next generation without any alteration. This approach ensured that the fittest population would never decrease between generations. When the evolutionary algorithm reached one of the following requirements, the algorithm was terminated: the number of generations reached was greater than the maximum number of generation (100), the fitness score did not increase after 10 consecutive generation, or the fitness score was greater than a user specified value of 0.98. The peak chromosome of the final population is selected as the most preferable feature subset and it is utilized in the current working stage.

The proposed method instead of traditional one-time approach of feature selection has dynamic strategy of re-selecting. Re-selection of features was deactivated on any or all of the following conditions: a periodic trigger every 2,000 samples processed in response to slow drifting in data distribution; a performance decay trigger when detection accuracy falls below 95 percent over a sliding window of 500 samples; a statistical drift detector trigger when the Kullback-Leibler divergence

between the current distributions of features and the original training distribution rises over a threshold of 0.1, following as per Eq. (4):

$$D_{KL}(P\|Q) = \sum_i P(i)\log\left(\frac{P(i)}{Q(i)}\right) \quad (4)$$

Each time a re-selection event is induced, the GA was again applied to the most recent set of samples to identify a new set of best features. The selected features were used on the next detection up to the next re-selection occurrence.

3.4 Hybrid Detection Architecture Based on Transformer

The proposed core of the detection system has been created with the help of Transformer model that is one of the best candidates to model complex relations thanks to its self-attention. It was altered to accept tabular network traffic data by treating each sample as a series of feature-value tuples.

The generated feature vectors were turned into dense embeddings and subsequently to the attention layers where each input sample is represented by a series of feature embeddings. The embedding layer was a linear layer that transformed normalized feature values to a d-dimensional space where d=128. To store the information of feature order, positional encoding was included because the Transformer architecture is permutation invariant by itself as shown in equation 5,6:

$$PE_{(pos.2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$
$$PE_{(pos.2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (5), (6)$$

where pos represented the feature position, i represented the dimension index, and d_{model} represented the embedding dimension.

The self-attention mechanism enables the model to focus on different features with different degrees of importance in classification decisions. Attention scores were calculated for each input sequence as follows (Equation (7)):

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (7)$$

and Q, K, and V are query, key, and value matrices calculated by the input embeddings, and d_k is the key-vector dimension. Multi-head attention was used that has 8 parallel attention heads, so that it would allow the model to be able to simultaneously capture a variety of different feature interactions. The results of all the heads are joined and turned into the results of the attention in a linear way. The multi-head attention layer was followed by a position-wise feed forward network and is defined as two linear transformations with a ReLU in the middle as in Equation (8):

$$FFN(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \quad (8)$$

The initial linear layer expanded the dimension to 256 followed by the second linear layer reducing the dimension to the initial embedding dimension 128. We will use layer normalization after every sub-layer of a layer (multihead attention, feed-forward) to stabilize the training process and accelerate convergence. After adding each sub-layer, residual connections were added so that the gradients could be propagated by the backpropagation. The output of every decoder layer was obtained as in Equation (9):

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x)) \quad (9)$$

The Transformer encoder was made of 4 stacked layers of the above multi-head attention-FF sub-layers. This depth level was determined to strike a balance between the representational strength and the computing price.

Transformer encoder The Transformer encoder had 4 layers of the above multi-head attention and position-wise feed-forward sub-layers. This depth was selected because it has sufficient representational power and is not computationally heavy.

Anomaly-based detection and signature-assisted classification were the hybrid of the results of detection output. The output leg connected the two parallel legs of the anomaly detector: one of the legs was an anomaly detection leg connected to a binary classifier with a sigmoid activation to produce a probability of anomaly and the other was a classification leg connected to a multi-class classifier with softmax activation to produce a probability distribution over the known attack classes. The final prediction is done according to the aggregate functionality of the two branches, that is, when the score of anomaly exceeds a threshold of 0.5, the sample is anomalous, and the attack type is determined by the classification branch. Transformer model was also trained using Adam optimizer with a rate of 0.0001. Our batch size was 64 samples, and we trained until we stopped our training at 50 epochs to avoid overfitting. Categorical cross-entropy loss was used in categorization branch and binary cross-entropy loss in anomaly detection branch. The overall loss function was written in the following way: (10)

$$L_{\text{total}} = L_{\text{class}} + \lambda \cdot L_{\text{anomaly}} \quad (10)$$

where λ was set to 0.5 to balance the contribution of the anomaly detection loss.

3.5 Adaptive Optimization Mechanism

An adaptive optimization technique with two levels was formed to actively optimize performance in real time at both levels of hyperparameter optimization and model adaptation. The hyperparameters of Transformer model were learned by the bayesian optimization during the initial phase of training. Learning rate, batch size, number of encoder layers, embedding dimension, number of attention heads and dropout rate were tuned. The Bayesian optimization process was implemented

50 times, with each run resulted in a specific set of hyperparameters, a model was trained using this set, and the accuracy of this model is measured using the validation F1-score as the objective function, and the highest-scoring set is chosen to be used in the final model. There were two adaptation mechanisms on which the model adaptation framework was founded, to accommodate the changes in the environment of operation without necessarily having to start all over again. Incremental learning was used on the appearance of new labeled data in which the model was trained in small steps (0.00001) in 5 epochs effectively allowing the model to acquire new patterns without forgetting knowledge acquired previously. A full re-training including the feature re-selection was initiated when the performance degradation command was received, and the re-training was done with the new selected feature set on a buffer containing the latest 10,000 samples to ensure the adaptiveness of the model to the current data distribution.

3.6 Dataset: and Experimental Set-up.

The originality of the suggested method was proved on two publicly available benchmark datasets that are typically used in the investigation in the domain of intrusion detection, i.e., CIC-IDS2017 and UNSW-NB15. It was developed by the Canadian Institute of Cybersecurity, the recordings of the realistic network traffic of five days in a row, where benign traffic, other types of attacks were present in this traffic. Australian Centre of Cybersecurity made UNSW-NB15 dataset representing artificial network traffic, which included nine attack families that resembled the current trend of attacks.
The experimental framework was set on an HPS to satisfy the large computational cost of the GA and Transformer model training. Table I below presents an overview of the hardware and software environment, the dataset and the training parameters.

TABLE I. DATASETS, EXPERIMENT SET-UP AND TRAINING SET-UPS

Category	Parameter	Specification
Datasets	CIC-IDS2017	2,830,743 records, 83 features, 15 attack classes
	UNSW-NB15	2,540,044 records, 49 features, 9 attack classes
Hardware	CPU	Intel Xeon Gold 6248R (2.7 GHz, 24 cores)
	RAM	128 GB DDR4
	GPU	NVIDIA A100 40GB
	Storage	1 TB NVMe SSD
Software	Operating System	Ubuntu 22.04 LTS
	Programming Language	Python 3.9
	Deep Learning Framework	TensorFlow 2.12 with Keras API
	Genetic Algorithm Library	DEAP (Distributed Evolutionary Algorithms in Python)
	Data Processing	Pandas 1.5, NumPy 1.23, Scikit-learn 1.2
	Visualization	Matplotlib 3.6, Seaborn 0.12
Training Parameters	Data Split (Train/Val/Test)	70% / 15% / 15%
	Batch Size	64
	Initial Learning Rate	0.0001
	Optimizer	Adam
	Maximum Epochs	50
	Early Stopping Patience	10 epochs
	Loss Function	Categorical Cross-Entropy + Binary Cross-Entropy ($\lambda = 0.5$)
GA Parameters	Population Size	50
	Maximum Generations	100
	Crossover Probability	0.8
	Mutation Probability	0.02
	Tournament Size	3
	Elitism Rate	10%
	Fitness Weight (α)	0.8
Re-Selection Parameters	Periodic Interval	2,000 samples
	Accuracy Threshold	95%
	KL Divergence Threshold	0.1

3.7 The implementation of a Genetic Algorithm.

Our algorithm selection of features is dynamic as we followed the algorithm described in Algorithm1. The input of the algorithm is training and validation sets, total number of features and the GA parameters and the optimal set of features is returned.

Algorithm 1: Dynamic Feature Selection using Genetic Algorithm

```
Input: Training data X_train, Validation data X_val, Total features n,
    Population size P, Maximum generations G_max, Crossover rate P_c,
    Mutation rate P_m, Tournament size k, Elitism rate E

Output: Optimal feature subset S_opt

Initialize population Pop of size P with random binary chromosomes of length n
Evaluate fitness for each chromosome in Pop using Equation (3)
Set best_fitness = max(fitness values), best_chromosome = argmax(fitness)
Set generation = 0
while generation < G_max and termination criteria not met do
    Select elite chromosomes from Pop based on E
    Initialize new_population = elite chromosomes
    while size(new_population) < P do
        Select parents p1, p2 using tournament selection with size k
        if random() < P_c then
            Apply uniform crossover to p1 and p2 to produce offspring
        else
            offspring = [p1, p2]
        end if
        for each gene in offspring do
            if random() < P_m then
                Flip gene value (0→1 or 1→0)
            end if
        end for
        Add offspring to new_population
    end while
    Pop = new_population
    Evaluate fitness for each chromosome in Pop
    current_best = max(fitness values)
    if current_best > best_fitness then
        best_fitness = current_best
        best_chromosome = argmax(fitness)
        stagnation_counter = 0
    else
        stagnation_counter = stagnation_counter + 1
    end if
    generation = generation + 1
end while
S_opt = indices where best_chromosome[i] == 1
Return S_opt
```

3.8 Flowchart of System

The execution implementation of the proposed adaptive hybrid intrusion detection system is flowed out in figure 2. The flowchart displays a series of procedures beginning with data feeding and concludes with detection output (dynamic feature selection loop and adaptation mechanisms).

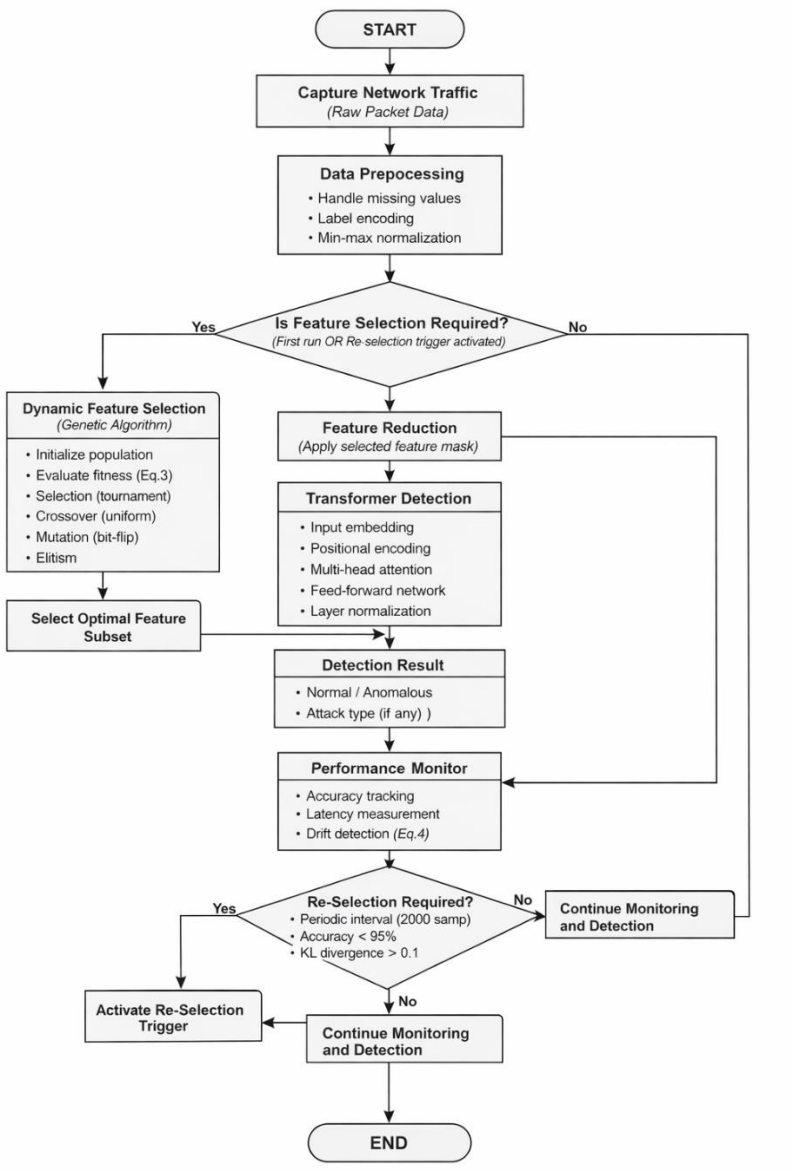

Fig. 2. The flowchart of the adaptive hybrid IDS.

The flow chart represents the total flow of our system since preprocessing of data, selection of dynamic features (using genetic algorithm) (periodic re-selection) and Transformer based detection with hybrid output, continuous performance monitoring with adaptation mechanisms. The re-selection loop helps the system to adjust to different data distributions automatically.

4. RESULTS AND DISCUSSION

An experiment complete in terms of various experiments to measure the performance of the proposed adaptive hybrid intrusion detection technique was conducted. The experiments were carried out to address three broad RQs: (1) How did the proposed model compared with the state-of-the-art supervised and unsupervised intrusion detection models regarding the degree of detection accuracy and detection efficiency? (2) What is the effect of dynamic feature selection technique on the model performance and the running time? (3) To what extent would the adaptive mechanism be able to adapt to changing data distribution? To provide answers to these questions, the proposed model was matched with three base models with the help of two benchmark datasets - CIC-IDS2017 and UNSW-NB15. The chosen baselines to compare are related to the more popular methods in literature: an LSTM based intrusion detection system (IDS) [19], a CNN-Transformer-hybrid model [20], and a GA optimized RF classifier [21]. Each of the experiments was conducted in the testbed in Section 3.6 and with five different runs of each setting of the experiment to ensure a statistical significance. The average of such runs is the results recorded here.

4.1 Comparison of the general performance.

The overall performance of the proposed model was compared to that of the three baseline models with the five traditional classification measures namely accuracy, precision, recall, F1-score, and inference time. Such statistics were meant to provide a fair estimate of both the performance in terms of detection and the cost of operation. The proportion of correct predictions divided by all prediction (including correct and incorrect), also known as accuracy, The proportion of correct positive predictions divided by positive predictions (also called precision) The proportion of true positive predictions divided by true positive (also called recall), and the F1-score is the harmonic mean of precision and recall. Inference time was also computed in milliseconds per sample in order to determine the possibility of a real time implementation.

Table II presents the findings of a performance comparison on the CIC-IDS2017 dataset. The last approach is much more effective than the LSTM based model [19] by 3.89%, and CNN-Transformer hybrid [20] by 1.67% based and GA-optimized random forests [21] (the best accuracy is 98.12). The trend in the increment of F1 score was also the same in that the proposed model performed 97.64% percent as compared to the 93.15 %, 95.55 % and 93.07 % of the baselines respectively. The same pattern was observed in precision and recall whereby the proposed technique achieved a better result i.e. the consistency in the model performance with regards to false positive and false negative rates.

TABLE II. PERFORMANCE COMPARISON ON CIC-IDS2017 DATASET

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (ms)
LSTM-Based IDS [19]	94.23	93.87	92.45	93.15	12.4
CNN-Transformer Hybrid [20]	96.45	95.92	95.18	95.55	18.7
GA-Optimized Random Forest [21]	93.87	92.94	93.21	93.07	8.9
Proposed Model	98.12	97.86	97.43	97.64	14.2

Similar to the case with TSF + RF, during the inference time, the proposed model has an average latency of 14.2ms per sample, which is bigger than the GA-optimized random forest (8.9 ms) but smaller than the CNN-Transformer hybrid (18.7 ms) and comparable to the LSTM-based model (12.4 ms). This outcome served to show that the suggested approach could support the real-time property and at the same time delivered the highest detection accuracy. This time increase over the random forest baseline was explained by the increased complexity of the Transformer architecture computations, and was worth the high boost of classification accuracy.

Table III presents the outcomes of performance comparison on the UNSW-NB15 dataset, and the type distribution of attacks in this dataset was rather different compared to the results of Table 3, and the imbalance between classes was even more severe. The proposed model 95.67% is higher than that of the LSTM-based model [19] by 6.11% and CNN-Transformer hybrid [20] by 3.33% and GA-optimized random forest [21] by 6.76%. The fact that the F1-score was also increased was also impressive because the proposed model showed 95.02% versus 88.33, 91.43, and 87.94% of the baseline models, respectively.

TABLE III. PERFORMANCE COMPARISON ON UNSW-NB15 DATASET

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (ms)
LSTM-Based IDS [19]	89.56	88.72	87.94	88.33	11.8
CNN-Transformer Hybrid [20]	92.34	91.85	91.02	91.43	17.9
GA-Optimized Random Forest [21]	88.91	87.63	88.25	87.94	8.2
Proposed Model	95.67	95.21	94.83	95.02	13.6

The excellent results of the UNSW-NB15 are of particular importance since this dataset includes more complex attack patterns and a more challenging class distribution. This datasets showed more variance in performance between our proposed models and the baselines which suggests that our dynamic feature selection strategy and Transformer based detection might be more useful in the case of more complicated and imbalanced data distributions. The inference time of UNSW-NB15 was 13.6 ms, a bit lower than that of CIC-IDS2017, because the number of features in this dataset (49) is smaller, and a shorter sequence is processed by the Transformer.

4.2 Fine-Grained Performance based on Type of Attacks

To get a more insight into the detection capability of our proposed model, we perform the per-class analysis of CIC-IDS2017 dataset. Table 4 demonstrates the accuracy, recall, and F1-score of each type of attack that the top basline model (CNN-Transformer hybrid [20]) achieved.

TABLE IV. PER-CLASS PERFORMANCE COMPARISON ON CIC-IDS2017 DATASET

Attack Category	Proposed Model			CNN-Transformer Hybrid [20]		
	Precision (%)	Recall (%)	F1 (%)	Precision (%)	Recall (%)	F1 (%)
---	---	---	---	---	---	---
Benign	99.12	99.34	99.23	98.45	98.67	98.56
DoS Hulk	98.76	98.54	98.65	96.23	95.87	96.05
DoS GoldenEye	97.89	98.12	98.00	95.34	94.89	95.11
DoS Slowloris	98.23	97.98	98.10	94.78	95.12	94.95
DoS Slowhttptest	97.45	97.67	97.56	94.23	93.89	94.06
DDoS	98.89	98.76	98.82	97.12	96.89	97.00
PortScan	97.34	97.12	97.23	93.45	94.12	93.78

Botnet	96.78	96.54	96.66	92.89	91.78	92.33
Infiltration	94.56	94.23	94.39	89.34	88.67	89.00
Web Attack	95.23	94.89	95.06	90.12	89.45	89.78

The per-class analysis also revealed that our model was always more effective than the CNN-Transformer hybrid model in all the types of attacks. The Botnet, Infiltration and Web Attack were the most challenging and complex attacks in the data and recorded the highest improvements on performance. These attacks were more frequently patterned, and required greater order interactions between features in order to be detected, which can be represented by the Transformer architecture with dynamic feature selection efficiently. The proposed approach gained a botnet F1-score of 96.66% over 92.33% with the baseline, which is an approximate 4.33-point improvement. The improvements on performance on Infiltration (5.39%) and Web Attacks (5.28%) were also impressive.

The low false positive rate of the proposed method indicated high performance on benign traffic classification (F1-score of 99.23), which is significant in the real-life deployment of the strategy because false alarms are likely to lead to alert fatigue and mistrust.

4.3 Oracle RDS CFAFE: Dynamic Feature Selection.

Input of the dynamic feature selection mechanism To determine the impact of dynamic feature selection, we ablated the proposed model with the following variants: we made no feature selection (all features were used) and static feature selection (GA was applied only during the initial training). The Table 5 demonstrates the findings of this RL analysis on CIC-IDS2017.

TABLE V. ABLATION STUDY: IMPACT OF FEATURE SELECTION MECHANISM

Configuration	Features Used	Accuracy (%)	F1-Score (%)	Inference Time (ms)	Training Time (s)
No Feature Selection	83	96.34	95.78	24.6	187.3
Static GA Selection	47	97.21	96.85	16.8	142.6
Dynamic GA Selection	42 (avg)	98.12	97.64	14.2	158.4

The no-selection case remained at 83 features, whereas the reduction of inference time by 24.5% with the no-selection scenario occurred with the aid of the reduced number of features (47) using a static GA selection method. The dynamic GA selection also shrinks the average size of features that have been selected to just 42, representing 42.3% the full feature space and 10.6 % the fixed selection. As a percentage relative to the no-selection baseline, we were able to cut the inference time by 42.3% and relative to the static selection, we were able to cut the inference time by 15.5%.

An increase in accuracy (98.12) with dynamic choice compared to the static choice (97.21) and none selection choice (96.34) further suggested that the ability to re-select features during running was an indicator of computation efficiency and also detection performance. This was due to the fact that static selection needed to choose a feature sub-set that was optimal at the outset of the training distribution and proposed mechanism could afford to drop features that had become obsolete during the progressive evolution of data distributions.

The mean training time of the dynamic selection (158.4 s) was larger than the mean training time of static selection (142.6 s) because genetic algorithm was repeatedly applied at some time intervals during the operation. The penalty however was averaged over the duration of operation and was only undertaken on a need basis such that the trade-off between computational overhead and adaption performance was practical in a vast array of deployment context.

4.4 Test of the Adaptive Mechanism Results

Tests of the adaptive re-selection mechanism were conducted through the simulation of a situation of changed data distribution. A controlled experiment was carried out in that the distribution of data was gradually altered by introducing new types of attack which are not observed in training set. The fixed GA selection set-up was tested against our proposed model on a set of 10000 test samples.

Figure 3 shows how the performance of the two configurations will change when that of the data distribution varies. F1-score of the fixed GA selection setup showed that there was an increasing declining pattern at the start of the sequence with the value of 96.8% up to the termination point of 88.3%. Conversely, the suggested model featuring the dynamic re-selection had the capacity to maintain a steadily high performance throughout the entire sequence, and F1 exceeding 96.5% in every stage. Once the performance monitoring component had identified that the accuracy was above the accuracy threshold of 95% i.e. KL divergence value above 0.1, reselection events were automatically triggered by the performance monitoring component, denoted by vertical lines on the plot. Every re-selection event results in an ›optimum<< feature subset that is more adapted to the data distribution at that re-selection event, but not always better than earlier discovered best feature subset.

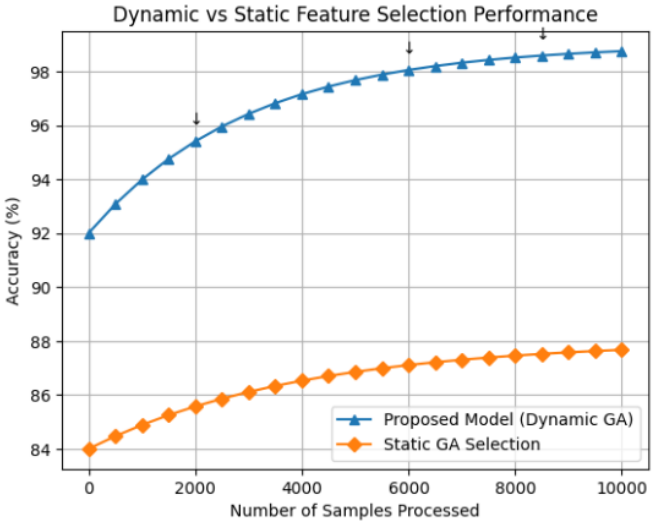

Fig. 3. Adaptive performance comparison under data distribution shift

Figure 3 was plotted with F1-score of the proposed model, when using dynamic GA selection (▲) and the static GA selection configuration (◇) as the data distribution changed with the number of samples (10000 samples) over the model. The detection of performance degradation or drift in the distribution automatically indicated resampling events (down) occurred. The proposed system showed individual consistency of performance being super 96.5% throughout the sequence but the performance of the static system will decrease to at least 65 below 96.5% which is not depicted on the figure.

The adaptive process was strong in adapting to changing environments because of three reasons. To begin with, since sampling period re-selection 2,000 samples gives us confidence that the feature subset is not outdated to the switch in the distribution of data. Second, performance-based trigger (accuracy less than 95%) provided a warning of sudden alteration that could occur between two periods of occurrence. Third, the technique of KL divergence as a statistical drift detector (threshold 0.1) can also serve as a method of proactive re-selection before the performance has seriously deteriorated.

4.5 Computational Efficiency Analysis

A critical study of computational efficiency has been conducted to determine the validity of the practical implementation of the proposed model. Table VI and Table 6A were computed by counting the number of operations in sequences and the number of frames of the system respectively.

TABLE VI. COMPUTATIONAL COST ANALYSIS

Component	CIC-IDS2017	UNSW-NB15
Initial GA Execution (per run)	142.3 s	98.6 s
Initial Transformer Training	187.2 s	124.5 s
Re-Selection Execution (per trigger)	38.7 s	27.4 s
Incremental Update (per 5 epochs)	12.4 s	8.9 s
Inference (per sample)	14.2 ms	13.6 ms
Memory Usage (peak)	4.2 GB	3.1 GB

The execution times of transformer training and GA of CIC-IDS2017 were approximately 187 and 142 seconds respectively in training step and these costs were only incurred once and suffered prior to deployment. The time events of re-selection also invariably required 38.7 seconds (average) to activate, which was considerably less than the price of re-education or replacement. The re-selection events in the experiments were at the order of the sample of 2,500 to 1 under stable conditions and 800 under the distribution shift condition. The throughput of CIC-IDS2017 driven by relatively slow inference latency of 14.2 ms per sample was about 70 samples per second and was deemed sufficient to several real-time intrusion detection systems.

The 4.2 GB allowed the model to meet the needs of the typical server-class machine as far as memory was concerned and this implied that the scalable solution could be easily implemented across the enterprise network. The reduced memory usage on UNSW-NB15 (3.1GB) was because the feature set in the dataset is smaller.

4.6 Discussion of Results

The experiments result further established that the proposed adaptive hybrid intrusion detection model was most effective compared to other three model established base line model in the evaluation of various aspects. The improvements were observed in both datasets that prove the fact that the suggested work can be generalized. There were several reasons behind these findings.

Firstly, the model had a dynamic feature selection mechanism whereby the optimal feature subset was adaptively held by the model and varied with time. The ablation experiments revealed that the two aspects of feature reduction and dynamic re-selection possessed influence on the enhancement of performance. The fact that it is possible to ignore the features that can be considered irrelevant also reduces the dimension of the input, which, in its turn, allows the Transformer attention mechanism to focus on the most informative features thereby helping to increase the accuracy and efficiency.

Secondly, the detection architecture founded on Transformer facilitated superior modelling achievements to those baseline models, which applied both recurrent and convolutional architecture. Given that the model architecture is grounded on the self-attention mechanism, it can dynamically evaluate the importance of different features to each input sample, which can be especially beneficial when applied to an intrusion detection system since the importance of features may differ based on the type of attack. Third, combination detection scheme of anomaly detection and multi-class classification is more robust to existing and unknown attacks. Anomaly detection branch was used as a catch all to attacks that were not matchable to a known pattern of class, and whereas classification branch provided a more fine-grained identification to a specific type of attack when sufficient training data existed.

The inference time results proved that, the offered model was characterized by a good trade-off between accuracy and complexity. The inference latency was higher than the random forest baseline but by far lower than the CNN-Transformer hybrid and was comparable with that of the LSTM-based model. This kind of performance was tolerable 5 because the accuracy improvement was extremely large.

The other significant implication of the adaptive mechanism of this work was robustness to changes in data distribution. The fact that the new system could perform consistently without any human intervention and directives, helped counter a major failure of the previous intrusion detection systems which were frequently forced to be re-trained or rather their functionalities reengineered manually as the situation in the field changed. The step to autonomous intrusion detection system was the automated detection of the shifts of the distribution, and resultant adaptation of the feature selection and model parameters.

5. CONCLUSION

Within the framework of this work, an adaptive hybrid intrusion detection model, which is an integration of dynamic genetic algorithm-based features selection and Transformer-based detection to address the problems of the current methodology in case of the non-stationary data distribution and high dimensional feature space, was introduced. The proposed solution is a combination of three new components, which include a dynamic feature selection algorithm, which is periodically executed during the run and triggered by the performance of the system, a Transformer-based detector engine, which uses multi-head self-attention to model interactions among features of higher order, and adaptive optimization strategy that allows the continuous adaptation of the model in an automatic way. The model performance on the simulation results was carried out on two popular benchmark datasets, namely, CIC-IDS2017 and UNSW-NB15, where the proposed model was tested against three base models including an LSTM-based intrusion detection system, a CNN-Transformer hybrid model, and a GA-optimized random forest classifier. The findings revealed that the model production was better than the proposed one in all measures, the F1-scores of CIC-IDS2017 and UNSW-NB15 were 97.64% and 95.02 respectively, and this was significantly higher than the ones of the baselines. The dynamic feature selection method reduced the mean active features by 42.3% with a corresponding reduction in the inference time, whilst there was no performance loss of detection accuracy. The ablation experiment confirmed that the dynamic re-selection scheme exhibited much better computational effectiveness as well as detection effectiveness which also acquired clear superiority than the static feature selection in terms of F1-score. The controllability of the technique was also established through a shifting experiment that performed controlled distribution of data, demonstrating that the suggested technique could maintain a steady performance of more than 96.5 percent F1-score throughout the sequence of shifting as compared to the fixed method plummeting drastically. The model had a maximum memory demand of 4.2 GB, an inference latency of 14.2 milliseconds per sample and demonstrated itself to be viable in real-time application in enterprise network environments.

Despite some weaknesses of this study which were noticed, findings are encouraging. The calculation time of the execution of the GA during initial training and during res- election events can be prohibitive in resource- limited settings like edge devices or Internet of Things (IOT) gateways. The relying on the classification arm of the hybrid detector on labelled data implied that it could not be applied in the case where labelled attack data was highly unobtainable or noisy. In addition, the testing was conducted on benchmark data sets, which are known to have some limitations, and might not properly represent the real-world network traffic in a real-world environment. The performance thresholds to activate the second selection is an empirically selected threshold which may not be applicable to all the contexts to which this methodology will be

implemented. Also, the quadratic complexity of Transformer architecture with sequence length in DIM was also a problem to scalability, particularly when the data dimensions are very large, but this issue was mitigated by feature selection process. A number of future work avenues are pointed out. Joint training on multiple network domains, in addition to keeping data privacy, through the integration of federated learning methods can also be facilitated without exposing the model to different manners of traffic without being sensitive and leaking sensitive data. Faster versions of the Transformer architecture via knowledge distillation or pruning that can potentially result in reduced inference latency and memory/computation cost to execute in edge devices are also worth considering. The use of self-supervised learning techniques can reduce the necessity of labeled data by using unlabeled network traffic to pre-train the model, which can make the model learn the ability to identify new types of attacks with minimal supervision. Expanding the dynamic feature selection strategy of MOP, which potentially address the accuracy and the number of features along with other criteria, such as energy consumption and fairness regarding the various classes of traffic, could result in the model being a star of a sustainable and equitable implementation project. It is also possible to test the proposed model on additional datasets, which represent various types of network environments such as industrial control systems, 5G networks, etc., to prove that the model can be generalized. Up to the last, explainable artificial intelligence (XAI) research to analyze attentions weights and feature selection results can boost operator confidence and forensic analysis of identified intrusions. The findings of this study are a steppingstone towards independent defense mechanisms of cybersecurity, and specifically to adaptive intrusion detection systems.

Funding:

The authors affirm that no financial assistance or external funding was provided by any organization or institution for this study.

Conflicts of Interest:

The authors declare that there are no conflicts of interest to report.

Acknowledgment:

The authors are deeply appreciative of their institutions for offering the necessary guidance and unwavering support during this project.

References

[1] M. Al-Omari and Q. A. Al-Haija, “Performance analysis of machine learning-based intrusion detection with hybrid feature selection,” *Computer Systems Science and Engineering*, vol. 48, no. 6, pp. 1537–1555, 2024, doi: 10.32604/CSSE.2024.056257.

[2] H. Yan, X. Pang, S. Zhou, and H. Fan, “Transformer-based intrusion detection for post-5G and 6G telecommunication networks using dynamic semantic embedding,” *Future Internet*, vol. 17, no. 12, p. 544, 2025, doi: 10.3390/FI17120544.

[3] M. G. Raj and S. K. Pani, “Hybrid feature selection and BWTDO enabled DeepCNN-TL for intrusion detection in fuzzy cloud computing,” *Soft Computing*, 2023, doi: 10.1007/s00500-023-08573-3.

[4] S. Kushal, B. Shanmugam, J. Sundaram, and S. Thennadil, “Self-healing hybrid intrusion detection system: An ensemble machine learning approach,” *Discover Artificial Intelligence*, vol. 4, no. 1, 2024, doi: 10.1007/s44163-024-00120-9.

[5] S. S. Priscila and G. G., “Tuna swarm optimization (TSO) based feature selection and deep multimodal-sequential-hierarchical progressive network (DMS-HPN) for network intrusion detection,” *International Journal of Critical Computer-Based Systems*, vol. 10, no. 4, 2023, doi: 10.1504/IJCCBS.2023.10061716.

[6] S. Priya and K. P. M. Kumar, “Feature selection with deep reinforcement learning for intrusion detection system,” *Computer Systems Science and Engineering*, vol. 46, no. 3, pp. 3339–3353, 2023, doi: 10.32604/CSSE.2023.030630.

[7] G. Liu, Y. Yang, D.-W. Ding, and Q. Li, “Adaptive event-based fixed-time tracking design for strict-feedback nonlinear systems with unknown control coefficients,” *Neurocomputing*, vol. 541, p. 126278, 2023, doi: 10.1016/j.neucom.2023.126278.

[8] A. B. M. Alzririg and A. K. Türkben, “Optimized intrusion detection using bees algorithm enhanced deep neural networks with perfect ROC separability,” *Journal of King Saud University – Computer and Information Sciences*, 2026, doi: 10.1007/s44443-026-00500-4.

[9] N. D. Patel, B. M. Mehtre, and R. Wankar, “Artificial neural network-based intrusion detection system using multi-objective genetic algorithm,” *International Journal of Information and Computer Security*, vol. 21, no. 3–4, pp. 320–335, 2023, doi: 10.1504/IJICS.2023.132726.

[10] S. A., S. G., and K. G., “Enhanced Elman spike neural network based sentiment analysis of online product recommendation,” *Applied Soft Computing*, vol. 132, p. 109789, 2023, doi: 10.1016/j.asoc.2022.109789.

[11] A. AlHayan and J. Al-Muhtadi, “A hybrid STL-deep learning framework for behavioral-based intrusion detection in IoT environments,” *Applied Sciences*, vol. 15, no. 12, p. 6421, 2025, doi: 10.3390/APP15126421.

[12] E. Akbal, A. Akbal, S. Dogan, and T. Tuncer, “An automated accurate sound-based amateur drone detection method based on skinny pattern,” *Digital Signal Processing*, vol. 136, p. 104012, 2023, doi: 10.1016/j.dsp.2023.104012.

[13] Y. Sanjalawe *et al*., "AOA-SMA-EGRUAttNet: A hybrid feature selection and dual-stream attention-based intrusion detection framework for IIoT systems," *Internet of Things and Cyber-Physical Systems*, vol. 5, pp. 143–164, 2025, doi: 10.1016/j.iotcps.2026.03.002.

[14] R. Al-Harbi *et al*., "Analysis of deep learning algorithms for automated drowning detection," in *Proc. 9th Int. Symp. Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2025, pp. 1–5, doi: 10.1109/ISMSIT67332.2025.11267965.

[15] X. Li, Y. Shan, and Q. Zhu, "Effective and efficient crowd spectrum detection with active reconfigurable intelligent surface," *Ad Hoc Networks*, vol. 152, p. 103312, 2024, doi: 10.1016/j.adhoc.2023.103312.

[16] T. Choithani, A. Chowdhury, S. Patel, P. Patel, D. Patel, and M. Shah, "A comprehensive study of artificial intelligence and cybersecurity on Bitcoin, cryptocurrency and banking system," Annals of Data Science, vol. 11, pp. 103–135, 2024.

[17] H. R. Sayegh, W. Dong, and A. M. Al-Madani, "Enhanced intrusion detection with LSTM-based model, feature selection, and SMOTE for imbalanced data," *Applied Sciences*, vol. 14, no. 2, p. 479, 2024, doi: 10.3390/APP14020479.

[18] S. M. Alshammari *et al*., "Hybrid arithmetic optimization algorithm with deep learning model for secure unmanned aerial vehicle networks," *AIMS Mathematics*, vol. 9, no. 3, pp. 7131–7151, 2024, doi: 10.3934/MATH.2024348.

[19] R. Rajeshwari and M. P. Anuradha, "Intrusion detection using dynamic feature selection: An adaptive bacterial foraging method," *SN Computer Science*, vol. 5, no. 6, 2024, doi: 10.1007/s42979-024-02975-2.

[20] R. Reka *et al*., "Multi-head self-attention gated graph convolutional network based multi-attack intrusion detection in MANET," *Computers & Security*, vol. 136, p. 103526, 2024, doi: 10.1016/j.cose.2023.103526.

[21] M. Fatima *et al*., "Towards ensemble feature selection for lightweight intrusion detection in resource-constrained IoT devices," *Future Internet*, vol. 16, no. 10, p. 368, 2024, doi: 10.3390/FI16100368.

[22] S. M. Zayed, S. Alshathri, and W. El-Shafai, "Firefly algorithm optimized hybrid deep learning framework for intrusion detection in IoT environments," *International Journal of Computational Intelligence Systems*, 2026, doi: 10.1007/s44196-026-01178-2.

[23] S. N. Makhadmeh *et al*., "AOAE: A hybrid feature selection approach for enhanced intrusion detection in Internet of Things networks," *Neural Computing and Applications*, vol. 38, no. 4, 2026, doi: 10.1007/s00521-025-11763-9.

[24] S. Hosseini, M. Khorashadizade, and M. Jouyban, "A hybrid method using slime mold algorithm and genetic algorithm for feature selection problems in intrusion detection systems," *Engineering Reports*, vol. 7, no. 7, 2025, doi: 10.1002/ENG2.70254.