ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2023.2.307

MENDEL Soft Computing Journal

A Hybrid Extreme Gradient Boosting and Long Short-Term Memory Algorithm for Cyber Threats Detection

Reham Amin✉, **Ghada El-Taweel, Ahmed Fouad Ali, Mohamed Tahoun**

Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, Ismailia, 41522, Egypt
reham_amin@ci.suez.edu.eg✉

Abstract

The vast amounts of data, lack of scalability, and low detection rates of traditional intrusion detection technologies make it impossible to keep up with evolving and increasingly sophisticated cyber threats. Therefore, there is an urgent need to detect and stop cyber threats early. Deep Learning has greatly improved intrusion detection due to its ability to self-learn and extract highly accurate features. In this paper, a Hybrid XG Boosted and Long Short-Term Memory algorithm (HXGBLSTM) is proposed. A comparative analysis is conducted between the computational performance of six established evolutionary computation algorithms and the recently developed bio-inspired metaheuristic algorithm called Zebra Optimisation Algorithm. These algorithms include the Particle Swarm Optimisation Algorithm, the Bio-inspired Algorithms, Bat Optimisation Algorithm, Firefly Optimisation Algorithm, and Monarch Butterfly Optimisation Algorithm, as well as the Genetic Algorithm as an Evolutionary Algorithm. The dimensionality curse has been mitigated by using these metaheuristic methods for feature selection, and the results are compared with the wrapper-based feature selection XGBoost algorithm. The proposed algorithm uses the CSE-CIC -IDS2018 dataset, which contains the latest network attacks. XGBoost outperformed the other FS algorithms and was used as the feature selection algorithm. In evaluating the effectiveness of the newly proposed HXGBLSTM, binary and multi-class classifications are considered. When comparing the performance of the proposed HXGBLSTM for cyber threat detection, it outperforms seven innovative deep learning algorithms for binary classification and four of them for multi-class classification. Other evaluation criteria such as recall, F1 score, and precision have been also used for comparison. The results showed that the best accuracy for binary classification is 99.8%, with F1-score of 99.83%, precision of 99.85%, and recall of 99.82%, in extensive and detailed experiments conducted on a real dataset. The best accuracy, F1-score, precision, and recall for multi-class classification were all around 100%, which does give the proposed algorithm an advantage over the compared ones.

Keywords: *Cyber Security, Intrusion Detection, Deep Learning, Feature Selection.*

Received: 18 November 2023
Accepted: 11 December 2023
Online: 17 December 2023
Published: 20 December 2023

1 Introduction

Since 2007, Deep Learning (DL) has become an increasingly important subfield of machine learning in various industries, including speech and pattern recognition, healthcare, vehicle design, image processing, financial forecasting, transportation, agriculture and legal applications, to name a few [16]. In addition, online Skype translation, automatic image captioning, and voice search (using technologies such as Amazon Alexa, Google Assistant, and Apple Inc. Siri) are some examples of successful DL applications [29]. Researchers and technology companies prefer to use DL to process big data effectively and efficiently [14, 36]. Large companies such as Facebook, Amazon, Google, and Microsoft are also using Deep Learning algorithms to analyze massive amounts of data on a daily basis. Artificial neural networks (ANN) are modeled after neurons in the human brain and are used in DL. The number of layers exhibited, which can be thousands, is referred to as deep [36].

There are many reasons why DL is well-liked. First of all, it can leverage large amounts of data for the best training and testing. For example, the ImageNet dataset of thousands of photos is easily accessible [31]. In addition to this the use of low-cost GPUs that can benefit from cloud services and are often used to train data [16]. With traditional methods of data analysis, it is difficult to properly understand the data and uncover important facts, even with machine learning algorithms [38]. Therefore, Deep Learning is used to dig deep into the network to obtain accurate and precise data.

The DL-based algorithm has many advantages when it comes to cyber threat detection. Security risks that are constantly changing include denial-of-service (DoS)

attacks, data spying, spoofing, and network resource occupancy, to name a few [2]. The traditional Deep Learning-based system is trained on a large dataset and then computes a variety of unique security threats. The system is then implemented in the real-world communications environment, where it can quickly detect the presence of hostile entities on the network in question [46]. All of these factors suggest that critical security measures are urgently needed to address these potential security issues, which DL can [25].

In addition to applying DL algorithms to real-world situations, this type of intrusion detection system assessment and tuning (IDS) uses realistic and modern datasets that depict both typical and anomalous network activity. In recent decades, several initiatives have been undertaken to create realistic datasets. Researchers have been able to adopt and evaluate more modern and optimistic datasets, although some of them have come under criticism for their shortcomings, lack of heterogeneity, limited availability, and critical level. It is believed that the selection of an appropriate dataset for intrusion detection plays a key role in ensuring the highly successful performance of Deep Learning-based IDs. Moreover, datasets are outdated due to the extensive nature of network attacks that have evolved significantly in recent years. To create an effective DL-based IDS, it is necessary to consider the current developments in IDS datasets [30].

The inclusion of a real traffic dataset, new and recent attack scenarios in CSE-CIC-IDS -2018 as opposed to recent datasets was one of several aspects considered in the selection of this dataset [8, 7, 33, 32, 2, 22, 26]. The CSE-CIC -IDS2018 dataset, while new, has been used to evaluate and compare various DL algorithms. It is available to the public at [13] and is very well organized. The dataset has been used extensively by the scientific community to benchmark cybersecurity because it contains a variety of attacks performed with different tools, arranged in a timeline, and combines regular and abnormal packet flows from the network [8]. In addition, the traffic was dynamically generated to mimic an enterprise network. The dataset also contains the original packet capture or PCAP files with each collected packet, in addition to the features collected from the network flows. This increases flexibility when using different pre-treatment and processing algorithms [5].

Numerous metaheuristic algorithms are available for use in feature selection. Metaheuristics can be classified into two main categories: single-solution algorithms and population-based algorithms [44]. Single solution-based algorithms may lead to local optima and prevent finding the global optimum. However, population-based algorithms are naturally able to avoid local optima [4, 39]. The population-based algorithms are classified into: evolutionary algorithms, swarm intelligence, bio-inspired algorithms, and physics based algorithms. Attempting each of these algorithms is challenging and time consuming. Therefore, we select the Zebra Optimization Algorithm (ZOA), a recently created bio-inspired metaheuristic algorithm, and six other long-standing algorithms from among them.

In this paper, a Hybrid XG Boosted and Long Short-Term Memory algorithm (HXGBLSTM) is proposed. Additionally, a comparative analysis is conducted between the computational performance of six established evolutionary computation algorithms and the recently developed bio-inspired metaheuristic algorithm called Zebra Optimisation Algorithm (ZOA) [41]. These algorithms include the Particle Swarm Optimisation Algorithm (PSO) [34], the Bio-inspired Algorithms (Bat Optimisation Algorithm (BOA) [17], Firefly Optimisation Algorithm (FOA) [12], and Monarch Butterfly Optimisation Algorithm (MBOA) [40], as well as the Genetic Algorithm (GA) [37] as an Evolutionary Algorithm. The dimensionality curse has been mitigated by using these metaheuristic methods for feature selection, and the results are compared with the wrapper-based feature selection XGBoost algorithm. The proposed algorithm uses the CSE-CIC -IDS2018 dataset, which contains the latest network attacks. XGBoost outperformed the other FS algorithms and was used as the feature selection algorithm.

Several authors compared different algorithms to evaluate the CSE-CIC -IDS2018 dataset. These deep impressions served as our motivation to thoroughly investigate the different deep learning algorithms used to analyze the CSE-CIC -IDS2018 dataset and tune the different crucial parameters that can be used to improve the DL algorithms and their learning processes on the dataset. Below is the main contribution of the paper:

- We have proposed a hybrid defensive algorithm called HXGBLSTM that combines the LSTM algorithm and the XGBoost algorithm for feature selection.
- We have compared six metahuristic algorithms against the proposed algorithm in order to select features for binary classification: Zebra Optimisation Algorithm (ZOA), Bat Optimisation Algorithm (BOA), Particle Swarm Optimisation Algorithm (PSO), Firefly Optimisation Algorithm (FOA), Monarch Butterfly Optimisation Algorithm (MBOA), and Genetic Algorithm (GA).
- We improved the accuracy of the proposed algorithm using two scenarios with the standard LSTM (i.e., before using FS) and the proposed HXGBLSTM (i.e., after using FS).
- We evaluated the efficiency of the proposed HXGBLSTM using the results of each classification mode (binary or multiclass). The proposed HXGBLSTM for cyber threat detection outperforms seven state-of-the-art deep learning algorithms for binary classification and four for multiclass classification when performance is compared.
- Our proposed algorithm proved promising when

we examined AUC, precision, recall, and the confusion matrix generated by each classification mode (CM) compared to other related algorithms.

The rest of this paper is organized as follows: the related work of the most popular and current DL algorithm applied to the CSE-CIC-IDS2018 dataset is provided in Section 2. Section 3 provides a detailed explanation of our proposed algorithm. Section 4 explains the experimental setup, parameter adjustment, the obtained results, and a comparison with other algorithms. Finally, the paper is concluded in Section 5, where the future perspectives are presented.

2 Literature Review

Hnamte et al. [23] proposed a dependable intrusion detection system using deep convolutional neural network. The framework's performance was assessed using significant factors like detection accuracy, false positive rate, and computational efficiency. ISCX-IDS 2012, DDoS (Kaggle), CICIDS2017, and CICIDS2018 are four publicly accessible IDS datasets that are used to assess the proposed system. The system achieved a detection accuracy range of 99.79% to 100%,

Alzughaibi et al. [6] proposed two Deep Neural Network (DNN) models for the CSE-CIC -IDS2018 dataset: one based on a Multi-Layer Perceptron (MLP) with BackPropagation (BP), the other on an MLP with Particle Swarm Optimization (PSO). They obtained results of 98.41% for multi-class classification and 98.97% for binary classification.

Wang et al. [45] analyzed network anomaly detection using the CSE-CIC -IDS2018 dataset in binary and multiclass classification. They tested several DL algorithms including DNN, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), LSTM, CNN RNN and CNN LSTM. The CNN RNN and CNN LSTM models achieved a maximum multiclass classification accuracy of 98.84% when compared to the individual DNN, RNN, CNN, and LSTM algorithms.

To analyze cyber threat data, the authors in [1] presented the Principal Component ANALYSIS (PCA) with DNN Model, which is based on DNN and PCA. By using DNN-PCA and only 12 features from the dataset, they were able to reduce the training time by half while maintaining the accuracy rate. 20% in testing and 80% in training. From the confusion matrix in the results, we can easily see the highest accuracy of 97.93%, along with a precision of 99.97% and a recall of 97.42%.

An innovative deep learning algorithm for behavior-based network intrusion detection was proposed by Antunes et al. [8] There are several hybrid algorithms are used such as CNN-PCA, CNN-Autoencoder, LSTM-PCA and LSTM-Autoencoder. The DDoS attack using the LSTM-PCA classifier provides the highest accuracy of 99.9%.

Al-Razib et al. [2] proposed a hybrid DNN-LSTM framework that is SDN-enabled to detect cyberattacks in smart environments. DNNLSTM, DNNGRU, and BLSTM were the classifiers tested. The highest accuracy, precision, recall, and F1 score for DNN LSTM were 99.55%, 99.36%, 99.44% and 99.42%, respectively.

In the Internet of Vehicles, Ullah et al. [42] suggested HDL-IDS: a hybrid deep learning architecture for intrusion detection. LSTM and GRU were combined with a DENSE ReLU layer in the suggested model. The outcomes revealed Precision of 0.9951, Recall of 0.996, F1-Score of 0.9952, and Accuracy of 0.9951.

Based on the CSE-CIC-IDS2018 dataset, Farhan et al. [18] provided a Performance analysis of intrusion detection using a deep learning algorithm. The LSTM has three layers, each having 78, 64, and 8 neurons, respectively. ReLu and Softmax are two of the activation methods that are used. The model could have been detected with an accuracy 99%.

Black Widow Optimized Convolutional Long Short-Term Memory (BWO-CONV-LSTM) Neural Networks on a MapReduce based platform was utilised by Kanna et al. [27]. The accuracy of the system for the CSE-CIC-IDS2018 dataset was 98.25%.

To improve learning from imbalanced samples, Zhang et al. [48] presented IoT intrusion detection based on data augmentation. The authors used ICVAE-BSM, which stands for Improved Conditional Variational Autoencoder and Borderline Synthetic Minority Oversampling Technique. The system obtained an accuracy of 98.67, an F1 score of 98.50, a recall of 98.95, and a precision of 98.04.

Kilincer et al. [28] proposed a thorough intrusion detection framework using boosting algorithms. The machine learning algorithms with default parameters were used to classify the newly created sub-datasets, while the Extra Tree algorithm selected the most optimal features for all datasets. The boosting algorithms include K-Nearest Neighbour (KNN), Naïve Bayes (NB), Decision Trees (DT), Multilayer Perceptron (MLP), Adaptive Boosting (AdaBoost), glioblastoma (GBM), Light Gradient Boosted Machine (LGBM), and eXtreme Gradient Boosting (XGBoost). The default settings were employed with five fold cross validation and achieved an accuracy of 99.94%.

For the CSE-CIC-IDS2018 dataset, Azeroual et al. [11] provided a framework for implementing the DL algorithm to Improve Intrusion Detection Systems (IDS). Using CNN, the model accuracy on validation data decreased to 83.55% after 50 iterations, but it returned a respectable accuracy of 92% after 30 iterations.

Assis et al. [10] presented a Gated Recurrent Unit (GRU) deep-learning system to protect against attacks in SDNs and achieved a 97% accuracy rate.

Deepdetect: Distributed Denial of Service (DDoS) threat detection using Deep Learning was presented by Asad et al. [9]. DNN (Deep Neural Network) with feed-forward backpropagation was used in the proposed model to evaluate the performance. 98% accuracy, an

F1 score of 0.99, and an AUC close to 1.

For effective detection of DDoS attacks in Software Defined Networks (SDNs), Haider et al. [21] presented a deep CNN ensemble system. Three 2-d convolutional layers with 128, 64, and 32 filters, respectively, form the ensemble CNN model, along with two maximum poolings, one flattening layer, and two dense FC layers. With Sigmoid as the output layer and ReLu as the activation function in the hidden layers, the system achieved 99.45% accuracy, 99.57% precision, 99.64% recognition, and 99.61% F1 score.

Ferrag et al. [32] examined deep learning algorithms for cybersecurity intrusion detection. Using CNN, Deep Boltzmann Machines (DBM), RNN, Deep Belief Networks (DBN), DNN, and Restricted Boltzmann Machine (RBM), they were able to achieve an accuracy of 97.28 to 97.37 percent.

Kim et al. [26] suggested a model of intrusion detection based on convolutional neural networks. The dataset was analyzed using vanilla RNN with 10 units and CNN with two layers. When applied to the same dataset, experimental results showed that the CNN model outperformed the RNN model.

3 The Proposed HXGBLSTM Algorithm

In this section, the details of the proposed Hybrid XG Boosted and Long Short-Term Memory (HXGBLSTM) algorithm is introduced. The main structure, the preparation and preprocessing of the dataset, and the assessment metrics are presented.

3.1 XGBoost Algorithm

Recursive feature elimination (RFE) is a sequential backward selection algorithm that is part of the Wrapper family. It uses a unique underlying method to pick features by recursively shrinking the feature set. Guyon I initially set up RFE based on the Support Vector Machine (SVM) model, which produced excellent results during the gene selection process [47]. All samples in the training set T is first included by the algorithm. The cross-validation is then carried out in T. The feature set size is then attempted to be decreased by iteratively creating new feature subsets. The prediction accuracy in each iteration is assessed using the Mean Absolute Error (MAE) which is shown in Equation (1), and the set of characteristics with the lowest average score based on MAE is chosen for the set of characteristics chosen. The features associated with the lowest feature importance are then removed after each iteration of feature subset computation and sorting of feature significance. The main structure of the XGBoost is shortened in Algorithm 1.

$$MAE = \sum_{i=1}^{n} \frac{|y_i - x_i|}{n} \tag{1}$$

where y_i is the prediction, x_i is the true value and n is number of samples.

Algorithm 1 XGBoost Algorithm.

Require: Training set T, which includes all samples
Ensure: Fold T into five folds using the cross-validation method
1: Build a new feature subset
2: **while** Feature subset is not empty **do**
3: Evaluate the prediction accuracy based on the Mean Absolute Error (MAE) as the objective function
4: Compute and order the IMportance $IM(x)$ of each feature x in the feature set using the Gain
5: Remove the feature with the lowest feature importance in the sequence's backward selection

All of the training samples from the training set are initially loaded into the algorithm. The cross-validation approach is then used to fold the training set into five folds. A new feature subset is created from these five folds in step 1. The loop begins iterating through the new feature subset in step 2, taking into account just the most significant features and disregarding others, until it becomes empty. The accuracy of the prediction is assessed in step 3 using the Mean Absolute Error (MAE) as the objective function. The importance of each feature x in the feature set is calculated and arranged in step 4. The feature with the lowest feature relevance in the sequence's backward selection is eliminated in step 5.

3.2 Long Short-Term Memory (LSTM)

Among the various DL algorithms, some of them, like DNN and CNN, are better suited to non-sequential data. They are capable of processing multimedia content. While being effective, they should not be used to identify patterns in time series data. For this reason, recurrent neural networks (RNNs) were developed. They are applied to activities such as speech recognition, image-to-text conversion, and video event detection [36]. RNNs have short-term memories that are based on repeating processes in hidden layers that relate to contextual data. Moreover, RNNs cannot function as long-term memory due to the problem with gradient vanishing and explosion [49]. Hence, a Long Short-Term Memory network (LSTM) for time-series prediction is proposed by Hochreiter and Schmidhuber in 1997 [24]. When memory cells are added to the hidden layer, the LSTM can govern how time-series data is stored in memory. A set of programmable gates (input, output, and forget gate) is used to transmit data between the cells in the hidden layer [19]. The vanishing gradient and explosion problem are avoided because LSTM can preserve the cell state through its gate mechanism, which can resolve both short-term and long-term memory dependency issues [20].

The basic LSTM cell in a memory cell with three gates—input gate, output gate, and forget gate—is depicted in Figure 1, which was previously explained in

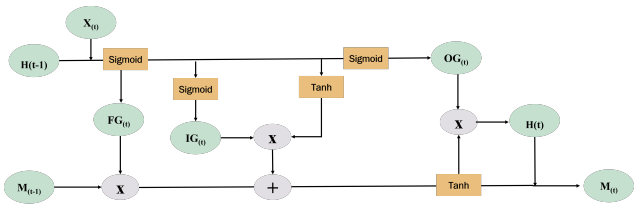


Figure 1: The block diagram of LSTM cell [20].

[20]. The input gate represented by the symbol IG is in charge of monitoring the most recent data within a memory cell. The input gate's value at time instance t is expressed in Equation (2).

$$IG(t) = sigmoid(W_{xi}X(t) + W_{hi}H(t-1) + B_i) \quad (2)$$

The output gate manages the distribution of the most recent data to other networks. It is denoted by the symbol OG and the output gate's value at time instance t is expressed in Equation (3).

$$OG(t) = sigmoid(W_{xo}X(t) + W_{ho}H(t-1) + B_o) \quad (3)$$

The forget gate determines whether or not the data should be erased based on the condition of the previous cell. It is denoted by the symbol FG and the forget gate's value at time instance t is expressed in Equation (4).

$$FG(t) = sigmoid(W_{xf}X(t) + W_{hf}H(t-1) + B_f) \quad (4)$$

The hidden state is depending on the cell memory state. It is denoted by the symbol $H(t)$ and the hidden state's value at time instance t is expressed in Equation (5).

$$H(t) = OG(t)\tanh(M(t)) \quad (5)$$

The cell memory state is denoted as $M(t)$ and its value at time instance t is expressed in Equation (6).

$$M(t) = FG(t) * M(t-1) + IG(t)* (sigmoid(W_{xc}X(t) + W_{hc}H(t-1) + B_c)) \quad (6)$$

The sigmoid function is expressed in Equation (7).

$$sigmoid(x) = \frac{1}{1+\exp^{-x}} \quad (7)$$

where X denotes the input vector, the weight matrices W_{xf}, W_{xi}, W_{xc}, and W_{xo} represent input weights, while W_{hf}, W_{hi}, W_{hc}, and W_{ho} represent recurrent weights. W_{hy} matrix of output weights; with the corresponding bias vectors B_f, B_i, B_c, B_o, and B_y.

Algorithm 2 HXGBLSTM Algorithm.

1: Initialize parameters Batch size, Epoch, Input shape($I_{0,d}$) and Load data instances from dataset
2: Apply data preprocessing
3: Apply XGBoost for feature selection
4: **for** Run = 1 to the number of runs (N) **do**
5: Define sequential model
6: **for** Epoch = 1 to the number of epochs (M) **do**
7: Detect and classify attacks using a fully connected Dense layer with neurons= 64 and activation = 'Tanh'
8: Compile the model using optimizer = 'Adam', loss function = 'Mean Squared Error'
9: Fit the model on the training set, validation set = 0.2, and batch size
10: Evaluate and predict model for the testing set
11: Obtain the overall best solution

3.3 The Structure of the Proposed HXGBLSTM

The block diagram of the proposed algorithm for improving the identification of cyber security threats is shown in Figure 2. One significant benefit is that it makes use of both LSTM as a DL method and XGBoost as a feature selection algorithm. Preprocessing the data, selecting the appropriate features, classifying the data, training and validating the algorithm, and lastly conducting testing and evaluation are the five main phases of the suggested approach. The proposed HXGBLSTM's primary structure is shortened in Algorithm 2.

The main steps of the proposed algorithm can be summarized as follows:

- **step 1**, parameters are initialized according to the values listed in Table 2.
- **step 2**, The files in the CSE-CIC-IDS-2018 dataset contain several data instances, which makes processing the data samples stored there and merging them to include each attack label extremely time-consuming and computationally challenging. Moreover, there is a high-class imbalance in the CIC-IDS-2017 and CSE-CIC-IDS-2018 datasets, which could indicate the system's poor accuracy and high false positive rate. For this reason, the data preprocessing phase is quite important. It is in charge of eliminating null or missing values, duplicate and inaccurate records, and the issue of class imbalance. The process of preparing the data for use in our proposed algorithm was explained in the following lines:
  - Elimination of NAN and infinite values: Any null, missing, or duplicate values are found and eliminated during this phase.
  - Label encoding: In this phase, each of the label's data is separated. The data associated with each label is then converted into a

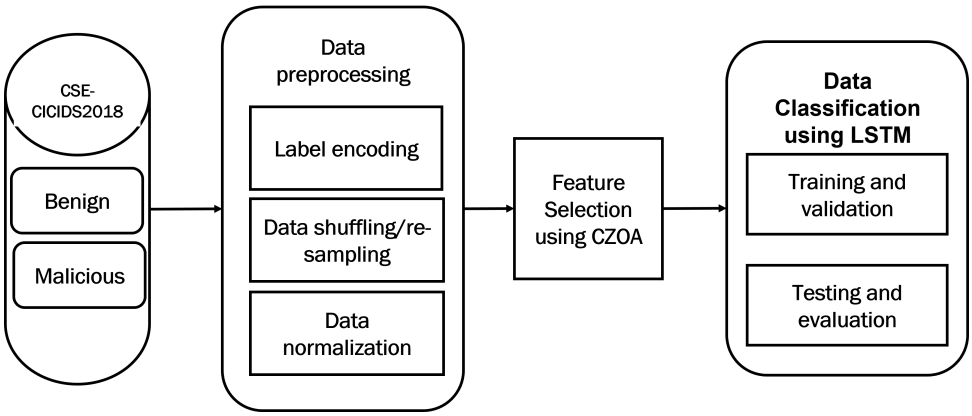


Figure 2: HXGBLSTM Block Diagram.

numerical matrix representation using label encoding. Target labels are additionally encoded with values ranging from zero to the number of classes minus one.

– Data shuffling/re-sampling: The CIC-CSE IDS 2018 dataset has an issue with class imbalance. The distribution of classes is significantly impacted by imbalanced datasets. Because of its higher prior probability, classification models tend to overclassify the larger class. Smaller class instances hence tend to be misclassified more often than larger class instances. This phase aims to address the problem of class imbalance. This can only be achieved by eliminating bias from the data and creating equal distributions, which are equal amounts of data from different attacks. The learning process is improved by avoiding biassed data and unbalanced samples. Therefore, resampling is used to equalize the data distribution across all categories and prevent overfitting, which occurs when one class has more samples than the others.

– Data normalization: This is the process of rescaling data from its original range to a new range ranging from zero to one where all values fall. Additionally, the values of all datasets have been scaled using the min-max normalisation approach to fall between zero and one. All values are divided by the greatest value that is encountered, or all values are divided by the range between the maximum and minimum values after the minimum value is eliminated. Assuming that x represents the data value that requires normalisation, y represents the normalised value of x, and that the lowest and highest values within the real range are min and max. Equation 8 normalises the x value.

$$y = \frac{x - min}{max–min} \tag{8}$$

- **step 3**, the proposed algorithm uses XGBoost for feature selection. With the 22 essential features indicated in Table 7, HXGBLSTM was retrained. Searching for the Uncorrelated List of Variables (SULOV) [43] serves as the foundation for XGBoost, and the features were selected. This algorithm determines which variable pair is more strongly related with the predefined correlation threshold.
- From **step 7** to **step 10**, it focuses on the creation of the X (predictor) and Y (target) variables. In this paper, a deeper LSTM network is proposed and consists of two input and output layers and three hidden layers. First, it uses the feature input layer to transfer an input layer to their representations. Afterward, the one-dimensional convolutional layer receives the sequence feed. Once the Tanh is used as an activation function, the outputs of the convolution layer are fed to max pooling layers. A dropout rate of 0.1 is applied when transferring the output to the LSTM layer. At last, the fully connected layers acquire and aggregate the data that was extracted from the LSTM layer. The final output is passed through a linear output layer for binary classification (0 for real, 1 for fraudulent), as well as multiclass classification. Training and testing sets of data were created via data splicing. While testing data is used to assess the model's performance on untested datasets, training data is used to train the DL algorithm. The data was split 80-20, which means that 80 percent was used for training and the remaining 20 percent was utilised for testing. Following that, performance is assessed using the evaluation metrics outlined in Section 4.3.

4 Experiments and Discussions

In order to validate the experimental findings, further tests with different parameters were run. Additionally, a comparison is made between the results of the standard LSTM with all features and the proposed HXGBLSTM algorithm with the selected features. Additionally, a variety of binary and multiclass classification modes were exam-

ined. This Section presents the experimental setups and findings. The description of the hardware and software configuration is given in Section 4.1. The CSE-CIC-IDS2018 Dataset, which is detailed in Section 4.2, was utilised to test the proposed HXGBLSTM. Performance is assessed using the evaluation metrics outlined in Section 4.3. The dataset was preprocessed and the parameters of the proposed algorithm were adjusted as explained in Section 4.4. In the initial setting of the standard LSTM, all 79 features from the dataset are fed into the model. However, in the subsequent tests, we employed XGBoost as a feature selection technique to reduce the total number of features and enhance the performance that was obtained. The feature selection phase of HXGBLSTM is described in Section 4.5. Both binary and multi-class classifications were used to assess the proposed HXGBLSTM algorithm. Section 4.6 discusses the efficiency of the proposed HXGBLSTM for binary classification, whereas Section 4.7 discusses the efficiency of the proposed HXGBLSTM for multi-class classification. Both Sections cover comparisons with other similar studies, the confusion matrix (CM) generated from the classification, and a detailed description of the performance obtained from each classification mode.

4.1 System Configuration

According to the resources for hardware and software acceleration for deep learning computing [50], a GPU working with NVIDIA CUDA (Compute Unified Device Architecture) is used to reduce computation times and possibly meet real-time data processing needs. Using a Kaggle Platform instance, we ran all of our studies on NVIDIA P100 and NVIDIA T4(x2) GPUs. We used Keras with a Tensorflow backend as our deep learning framework. The NVIDIA P100 has the Pascal architecture, 16GB of RAM, 9.5 TeraFLOP/s of Single Precision FLOPs, and 732 GB/s of memory bandwidth. The NVIDIA T4, on the other hand, has 16GB of memory, 8.1 TeraFLOP/s of Single Precision FLOP/s, and 320 GB/s of memory bandwidth. For CPU tasks, the 16GB RAM allocation is raised to 30GB per session there.

4.2 CSE-CIC-IDS2018 Dataset

The CSE-CIC-IDS2018 intrusion detection dataset was created in 2018 by the Communications Security Establishment and the Canadian Institute for Cybersecurity, both of which have their headquarters in Fredericton, Canada [**?**]. The most recent intrusion dataset, acquired to conduct real attacks, is the CSE-CIC-IDS2018 [3]. It is available to the public. CSE-CIC-IDS2018 has a larger capacity than CICIDS2017, with over

Table 1: Labels for CSE-CICIDS2018 Attacks.

Label	attack name
1	FTP-BruteForce
2	SSH-Bruteforce
3	DDOS attack-HOIC
4	Bot
5	DoS attacks-GoldenEye
6	DoS attacks-Slowloris
7	DDOS attack-LOIC-UDP
8	Brute Force-Web
9	Brute Force-XSS
10	SQL Injection

400GB. The dataset includes both the attack dataset's required standards and a broad range of attacks. It primarily comprises seven distinct attack scenarios: distributed denial-of-service, botnet, denial-of-service, brute force, heartbleed, web attacks, and network infiltration [13]. A dataset was created using the HTTP, HTTPS, FTP, SSH, and email protocols to simulate the online activities of 25 fake users. The sub-datasets that make up CIC-2018 were collected over a period of 10 different days, following the injection of 16 distinct forms of attacks.

There are several attacks, including SSH-BruteForce, FTP-BruteForce, Brute Force -XSS attack, Brute Force -Web, SQL Injection, DoS attacks using Hulk, SlowHTTPPTest, Slowloris, DoS attacks using GoldenEye, DDOS attacks using HOIC, DDOS attacks using LOIC-UDP, DDOS attacks using LOIC-HTTP, and more. In Table 1, the attacks are listed. To more accurately represent the attacks, a network infrastructure with machine diversity similar to real-world networks was developed (five departments making up the victim organisation, with fifty attacker machines, four hundred victim machines, and thirty servers) [3, 2]. The dataset includes 80 features that CICFlowMeter-V3 retrieved from the traffic, together with forward and backward collected system logs and network traffic for each machine [33]. Ten files totaling $15,450,706$ rows, each with 80 characteristics, make up the CSE-CIC-IDS2018 dataset [26].

Table 2: Adjusted hyperparameters of the proposed HXGBLSTM.

Paramter	**Value**
Optimizer	Adam
Learning rate	0.001
Hidden Nodes	64
Dropout rate	0.1
Batch size	32
Epochs	50
Activation function	Tanh
Loss function	Mean Squared Error

4.3 Evaluation Metrics

Commonly used performance metrics like accuracy, precision, recall, AUC value, and F1 are used to evaluate classifiers based on our proposed algorithm [30]. Furthermore, trials were conducted extensively to distinguish between malicious and legitimate records, so the confusion matrix (CM) was used to compute the performance metrics in our work [15, 8].

The proposed algorithm was assessed using the following metrics:

– Accuracy (Acc) is the percentage of all expected occurrences, whether normal or abnormal, that were correctly predicted to all observations. This metric, which is commonly used to evaluate model performance, is especially useful when the classes are not balanced. Its value is calculated using the Equation 9.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

– Precision is the number of accurately predicted positive observations relative to all expected positive observations. Low false positive rates are negatively correlated with accuracy. The greater precision levels correspond to better results. Its value is calculated using the Equation 10.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

– Recall is the ratio of correctly predicted positive observations to all observations, and it should be as high as possible. Its value is calculated using the Equation 11.

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

– F1 score is produced by harmonically averaging Precision and Recall. Its value is calculated using the Equation 12.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{12}$$

– Area Under the Curve (AUC) indicates how effectively a machine learning model matches our expectations for identifying or classifying various scenario types [35]. It indicates the probability that a positive sample will outnumber a negative sample when rating is taken into consideration. Its value is calculated using the Equation 13.

$$AUC = \frac{n}{P.N} \tag{13}$$

The components of the confusion matrix are shown in Figure 3. The following acronyms can be used to refer to the CM:

Confusion Matrix	Actual	
Predicted	TP	FP
	FN	TN

Figure 3: Confusion Matrix.

- TP: The model accurately categorises benign events.
- TN: The model accurately identifies malicious attempts.
- FP: In all observations, anomalies are incorrectly expected to be normal occurrences. This number should ideally have a low value.
- FN: Malicious attacks are incorrectly classified by the model as benign occurrences. This number should ideally have a low value.

4.4 Parameter Setting

Setting the different parameters was harder than it seemed at first. At this point, we decide not to use any algorithms for hyperparameter optimization. However, when the outcomes weren't satisfactory, we manually made a random adjustment at first. Afterthat, We proceed step-by-step to manual adjustment.

Using the default hyperparameter settings and without making any parameter adjustments, the dataset was split 80-20 into 80% and 20% training and testing sections, respectively. For a fair evaluation, five cross-validation splits are performed, and the mean is computed. The obtained accuracy was 97.78 percent, the loss was 0.01; the precision was 97.88; the recall was 97.65; and the AUC was 98.72 . However, in contrast to other comparable research, that was low.

In this step, the hyperparameters of the proposed algorithm were randomly adjusted in an attempt to improve performance. For five cross-validation splits, the suggested approach was evaluated using the same performance metrics. The suggested algorithm's default settings were randomly altered for thirty epochs in each run to see if it affected the outcomes.

The first random adjustment was done when the hidden nodes were 256 and the filters were 64. This led to an accuracy of 97.78%, F1 of 97.73, a precision of 97.88, recall of 97.65, and an AUC of 98.72. Next, several random adjustments are implemented for the proposed algorithm, including the following:

- While hidden nodes were extended to 128 and filters to 512, accuracy decreased from 97.78 to 97. The results of HXGBLSTM are 97.5 percent AUC, 97.53% F1, 97.54% precision, and 96.48% recall.
- After changing the number of hidden layers and reducing hidden nodes 128—a drop in accuracy from 97.78 to 96.82 occurred. It achieves 96.7 F1, 96.7 precision, 96.79 recall, and 98.23 AUC.
- By halving the number of filters and hidden nodes, the accuracy drops from 97.78 to 95. The precision

Table 3: The results of the sequential adjustment of the dropout rate for the HXGBLSTM algorithm.

Adjusted Parameters			Performance metrics					
Dropout	Hidden nodes	Activation Function	Accuracy	Loss	F1	Precision	Recall	AUC
0.0	64	relu	94	0.01	97.82	97.88	97.83	98.81
0.1	**64**	**relu**	**95**	**0.01**	**97.38**	**97.74**	**97.07**	**98.42**
0.2	64	relu	93	0.01	96.78	97.32	96.47	98.1
0.3	64	relu	89	0.02	87.93	97.66	83.68	91.75
0.4	64	relu	88	0.02	89.98	95.06	87.6	93.49
0.5	64	relu	90	0.02	96.53	97.85	95.44	97.61
0.6	64	relu	86	0.03	86.55	98.54	81.45	90.66
0.7	64	relu	85	0.04	92.07	96.88	89.62	94.65
0.8	64	relu	78	0.04	85.9	99.35	79.12	89.53
0.9	64	relu	60	0.06	27.33	36.36	27.3	63.65

is 97.15, the recall is 96.11, and the AUC is 97.91. The F1 is 96.53.

After the random adjustment produced unsatisfactory results, this step switched to sequential manual adjustment to see if altering the hyperparameters had an impact on the HXGBLSTM algorithm's performance. The accuracy of the system is influenced by various parameters, such as the activation function, number of hidden nodes, and dropout rate. We proceed with modifying each of these elements separately. The most appropriate parameters for evaluating the HXGBLSTM were identified after several runs, and the outcomes were contrasted with those of the earlier experiments.

The results of several in-depth tests conducted with various hyperparameters for the IZOA algorithm are shown in Tables 3, 4, and 5 in where the overall best values are reported in bold text.

4.4.1 The Adjustment of the Dropout Rate

As seen in Table 3, the HXGBLSTM algorithm's dropout rate is updated progressively from 0.1 to 0.9. In each trial, the dropout rate is increasing by 0.1 and using fixed values for hidden nodes of 64 and the relu as an activation function. Bold text reports the overall best values. It has been observed that accuracy increases from 60% to 95% when the dropout rate is decreased from 0.9 to 0.1. The results clearly show that the ideal value for the dropout is 0.1, yielding the maximum accuracy of 95%.

4.4.2 The Adjustment of the Hidden Nodes

The number of hidden nodes for the HXGBLSTM algorithm is adjusted consecutively from 8 to 128 while maintaining stable values for a dropout rate of 0.1, as it was the ideal value. To perform this update, each trial's hidden node count is doubled. The relu served as an activation function throughout these upgrades. The performance changes while updating the hidden nodes are shown in Table 3. Reducing the number of hidden nodes from 8 to 64 has been found to increase accuracy from 61.69% to 97.74%. Thus, 64 is the best value for the hidden nodes, and when the other parameters are fixed, this gives the highest accuracy of 95%.

4.4.3 The Adjustment of the Activation Function

Every trial, the activation function for the HXGBLSTM method is updated using a combination of tanh, sigmoid, softmax, linear, and relu. The dropout rate of 0.1 and the number of hidden nodes of 64 were fixed because these were the best values, as indicated in Tables 3 and 4, respectively. Based on the best results obtained, Table 5 displays the performance progress in order during updates. It is observed that increasing the accuracy from 93.92% to 98.23% occurs when the activation function is changed from relu to tanh. The data shown clearly shows that, while other parameters are fixed, tanh is the optimal choice for the activation function, yielding the highest accuracy of 98.23%.

Table 2 displays the optimised adjusted hyperparameters for the HXGBLSTM algorithm. Using the Adam optimizer, the trainable parameter was changed at a learning rate of 0.001. Furthermore, it's clear that performance changes gradually as one advances from using the default hyperparameters, which produce an accuracy of 90%, to adjusting the hidden nodes to 64, which produces an accuracy of 96.74%, to enhancing the dropout rate to 0.1, which produces an accuracy of 95.23%, and lastly choosing tanh as an activation function, which produces an accuracy of 98.23%.

4.4.4 The Adjustment of the Epochs

After selecting the best settings, the next trial was advanced by increasing the number of epochs in each iteration. The epoch count increased from 30 to 50, and the number of runs increased from 5 to 20. The top performing results overall are shown in Table 6. This table shows the improvement in the binary classification performance for HXGBLSTM based on Accuracy, Loss, F1, Precision, Recall, and AUC throughout several runs and epochs. Performance is enhanced by increasing the number of runs and epochs. When the

Table 4: The results of the sequential adjustment of the hidden nodes for the HXGBLSTM algorithm.

Adjusted Parameters			Performance metrics					
Dropout	Hidden nodes	Activation Function	Accuracy	Loss	F1	Precision	Recall	AUC
0.1	8	relu	61.69	0.05	41.41	62.94	37.53	68.74
0.1	16	relu	77.59	0.03	69.32	86.11	61.05	80.41
0.1	32	relu	90.93	0.02	94.35	95.3	93.98	96.74
0.1	**64**	**relu**	**97.74**	**0.01**	**96.04**	**97.02**	**95.36**	**97.53**
0.1	128	relu	96.55	0.01	97.61	97.74	97.54	98.66

Table 5: The results of the adjustment of the activation function for the HXGBLSTM algorithm ranked by the best performance.

Adjusted Parameters			Performance metrics					
Dropout	Hidden nodes	Activation Function	Accuracy	Loss	F1	Precision	Recall	AUC
0.1	**64**	**tanh**	**98.23**	**0**	**0.994711**	**0.994847**	**0.994643**	**0.997061**
0.1	64	sigmoid	97.85	0	0.994694	0.99507	0.994383	0.996942
0.1	64	softmax	97.15	0.01	0.984358	0.986511	0.982971	0.990785
0.1	64	linear	95.62	0.01	0.979264	0.990119	0.969471	0.984243
0.1	64	relu	93.92	0.01	0.958432	0.964293	0.955074	0.975615

Table 6: The efficiency of accuracy through a different number of epochs of the proposed algorithm based on binary classification.

# Epochs	**# Runs**	**Acuracy**	**Loss**	**F1**	**Precision**	**Recall**	**AUC**
20	5	97.78	0.01	97.73	97.88	97.65	98.72
30	5	98.65	0	98	98.6	97.6	98.7
	10	98.95	0	99.4	99.4	99.4	99.7
50	**20**	**98.98**	**0**	**99.2**	**99.4**	**99.2**	**99.5**

number of epochs and runs is increased, the accuracy averages out at 98.98 instead of 98.23 as obtained after adjusting parameters.

Prior to FS, the HXGBLSTM was assessed and yielded the following results: 97.78% accuracy, 97.73 F1, 97.88 precision, 97.65 recall, and 98.72 AUC. However, the accuracy increased to 98.65%, the precision to 98.6, the recall to 97.6, the AUC to 98.7, and there was no loss. This is achieved by using five different runs, thirty epochs, and the adjusted hyperparameters.

After adjusting the hyperparameters to 30 epochs and 10 distinct runs, the accuracy reached 98.95%, the F1 was 99.4, the precision was 99.4, the recall was 99.4, the AUC was 99.7, and there was no loss. The results demonstrated 98.98% accuracy, 99.2 F1, 99.4 precision, 99.2 recall, 99.5 AUC, and zero loss using 50 epochs and 20 separate runs.

4.5 Feature Selection in HXGBLSTM

We concluded that the optimum performance would come from not utilising all of the features in the dataset because some of them were redundant and unnecessary. This provides a motivation to investigate whether using the optimal features has an impact on performance. In order to improve the result and decrease the amount of features, we thus used the feature selection algorithm.

The stronger features were successfully chosen using the XGBoost FS algorithm, which eliminated the weaker ones. A thorough explanation of the main algorithm used by XGBoost may be found in Section 3.1. Just 22 out of the 79 features in the original dataset were selected using the XGBoost FS algorithm. Using the 22 important features given in Table 7, the proposed HXGBLSTM algorithm was further improved.

Table 7: Overview of the 22 selected features based on XGBoost feature selection algorithm.

Bwd Pkt Len Mean	Fwd Header Len	Bwd IAT Mean	Dst Port	Fwd PSH Flags
Init Fwd Win Byts	Init Bwd Win Byts	Fwd Seg Size Min	Subflow Bwd Pkts	Flow IAT Mean
Flow Pkts/s	Flow Byts/s	Flow IAT Std	URG Flag Cnt	Bwd Header Len
Fwd Pkt Len Max	SYN Flag Cnt	Flow Duration	PSH Flag Cnt	RST Flag Cnt
Tot Fwd Pkts	Protocol			

The algorithm's goal during the FS process is to maximise memory usage with a 0.7 correlation limit. The memory utilisation is reduced by 62.3 percent. The feature selection process went through multiple iterations. Each time around, a few of the less important features were removed.

In total, there are 79 features. In the initial iteration, ten ID or low-information features are removed. It then retains unnecessary variables out of the next processing, leaving 68 features. The next step involves searching for uncorrelated lists of variables (SULOV) using the 68 features. Next, the variables that show high correlation are eliminated.

Using SULOV, 35 characteristics were ultimately

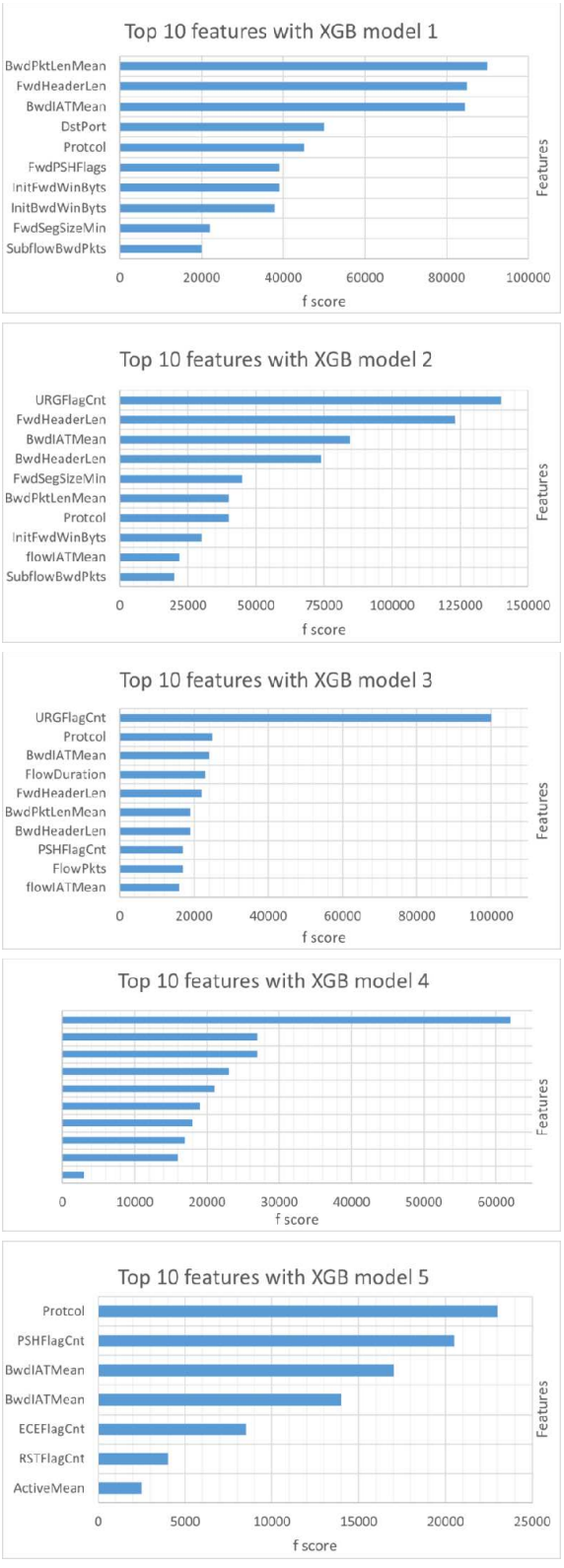


Figure 4: The cyclical development of the HXGBLSTM FS algorithm.

chosen. Every iteration of the XGBoost FS technique chooses the top 10 features from the range of selected feature in a recursive manner. The XGBoost FS completed in 24 seconds, however the total FS took 186 seconds to complete. Finally, the result included a list of 22 significant features. The HXGBLSTM FS algorithm's cyclical development is shown in Figure 4.

Table 8: Efficiency of the proposed HXGBLSTM for binary Classification.

Feature Selection Algorithm	Accuracy	Loss	F1	Precision	Recall	AUC
STD LSTM	93.31	5.9	93.6	93.72	93.61	93.61
ZOA	97.34	2.3	97.47	97.54	97.47	97.47
BOA	99.3	1	99.33	99.33	99.33	99.33
PSO	99.31	1	99.37	99.38	99.37	99.37
GA	99.37	1	98.73	98.73	98.73	98.73
MBOA	99.4	1	99.48	99.48	99.49	99.49
FOA	99.55	0.3	99.7	99.71	99.7	99.7
HXGBLSTM	**99.8**	**0**	**99.83**	**99.85**	**99.82**	**99.9**

4.6 Efficiency of The Proposed HXGBLSTM for Binary Classification

The purpose of this experiment is to assess the proposed HXGBLSTM's binary classification performance. Network traffic is classified as normal or abnormal by the binary classification(0 for normal, 1 for fraudulent). This was done using the optimised adjusted hyperparameters for the HXGBLSTM, which are listed in Table 2. The preprocessed CSE-CICIDS2018 dataset was used for this.

4.6.1 Performance of The Binary Classification of The Standard LSTM and The Proposed HXGBLSTM

The performance of the proposed HXGBLSTM is compared to that of the standard LSTM. The efficiency of the proposed HXGBLSTM for binary classification in comparison to the standard LSTM is shown in Table 8. The best results in this table are indicated in bold text. This table shows a comparative analysis between the computational performance of six established evolutionary computation algorithms and the recently developed bio-inspired metaheuristic algorithm called Zebra Optimisation Algorithm (ZOA). These algorithms include the Particle Swarm Optimisation Algorithm (PSO), the Bio-inspired Algorithms (Bat Optimisation Algorithm (BOA), Firefly Optimisation Algorithm (FOA), and Monarch Butterfly Optimisation Algorithm (MBOA), as well as the Genetic Algorithm (GA) as an Evolutionary Algorithm. When the features for binary classification were selected, the outcomes were compared to the proposed algorithm. The wrapper-based feature selection XGBoost algorithm has been demonstrated to perform better than the other FS algorithms. The CSE-CICIDS2018 dataset was utilised to obtain these results, which are the average of twenty runs, each with fifty epochs. This table demonstrates how effectively the proposed HXGBLSTM algorithm employed, with an accuracy of 99.80%, loss close to zero, an F1 score of 99.83, a precision of 99.85, a recall of up to 99.82, and an AUC of 99.9. Figures 7a 7b, 8a, and 8b displayed the confusion matrix and ROC for a more detailed look at the results of the experiment.

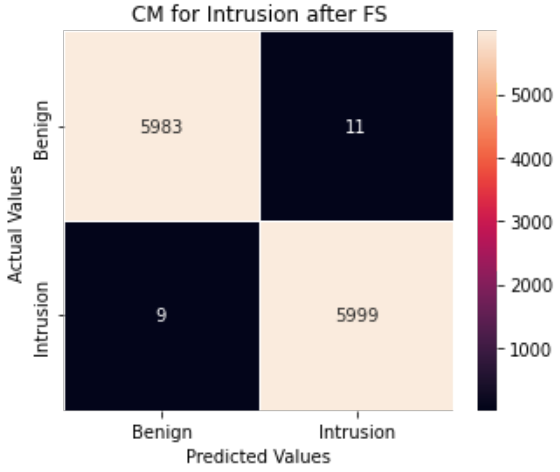


Figure 5: Performance of binary classification for the standard LSTM and HXGBLSTM.

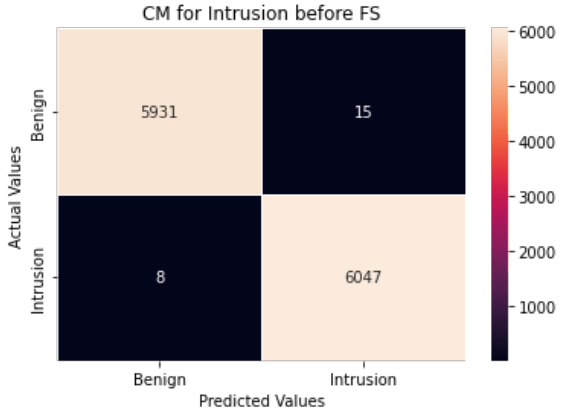


Figure 6: Performance of the standard LSTM based on binary classification.

4.6.2 HXGBLSTM Vs. Other Algorithms

To accurately evaluate the effectiveness of HXGBLSTM, we conducted in-depth comparison with the most recent DL algorithms that were employed in the cyber threat detection literature. In this comparison, we used deep learning algorithms from earlier studies such as [11], [28], [10], [32], [1], [9], BWO-CONV-LSTM [27], [48], [45], [6], [18], [21], [42], and [1]. The results of the comparison for deep learning approaches are shown in Table 9. Bold text in Table 9 represents the overall best solution. The proposed HXGBLSTM's achieves an accuracy of 99.80, F1 of 99.83, the precision of 99.85, recall of 99.82, loss of zero, and AUC of 99.9. These results show that, in terms of the overall performance of the binary classification, our proposed algorithm performed better than other algorithms.

4.7 Efficiency of the Proposed HXGBLSTM for Multi-Class Classification

This experiment aims to multiclassify network traffic into 10 types of attacks (as listed in Table 1) and the benign class in order to assess the efficiency of the proposed HXGBLSTM for cyber threat detection. The HXGBLSTM's optimised adjusted hyperparameters, which are given in Table 2, were used for this. HXGBLSTM's performance is evaluated by comparing it with the most recent research and utilising the CSE-CICIDS2018 dataset.

Table 9: Comparison with Related Literature Based on Binary Classification.

Ref.	Accuracy	F1	Precision	Recall
[1]	97.93	98.68	99.97	97.42
[48]	98.67	98.5	98.04	98.95
[45]	98.85	98.83	98.85	98.85
[6]	98.97	99.38	**99.98**	98.8
[21]	99.45	99.61	99.57	99.64
[42]	99.51	99.52	99.51	99.6
[2]	99.55	99.42	99.36	99.44
HXGBLSTM	**99.8**	**99.83**	99.85	**99.82**

4.7.1 Performance of the Multi-Class Classification of The Standard LSTM and The Proposed HXGBLSTM

The purpose of this experiment is to use the CSE-CICIDS2018 to investigate how well HXGBLSTM performs in correctly identifying ten different types of attacks. For each attack type, the accuracy, loss, F1 score, precision, recall, AUC, and CM were assessed. Additionally, a comparison was made between the outcomes of HXGBLSTM and the standard LSTM To guarantee an equitable comparison, the two algorithms— HXGBLSTM and the standard LSTM —were implemented and run in the same setting. The efficiency of the proposed HXGBLSTM for multiclass classification in comparison to the standard LSTM is shown in Table 10. Furthermore, Figures 7, and 8 display the confusion matrix based on multiclass classification for the standard LSTM and HXGBLSTM respectively.

It is evident from these outcomes that HXGBLSTM outperformed other related algorithms across the board for every evaluation criteria. This is because the suggested HXGBLSTM algorithm incorporates both XGB and LSTM.

With the standard LSTM, the accuracy for the FTP-BruteForce attack was 99.12, the loss was zero, the F1 score reached 99.5, the precision scored 99.52, the recall achieved 99.49, and the AUC attained 99.72. For the SSH-Bruteforce attack, the accuracy was 99.77, the loss was zero, the F1 score reached 99.8, the precision scored 99.8, the recall achieved 99.8, and the AUC attained 99.8. According to DDOS attack-HOIC attack, the accuracy was 99.89, the loss was zero, and the F1 score, precision, recall, and the auc attained 99.95.

For the Bot attack, the accuracy was 99.58, and the loss raised to 0.33, while the F1 score, precision, recall, and AUC were fixed at 99.56. Regarding the DoS attacks-GoldenEye attack, accuracy reached 99.87, the loss still zero, while the F1 score, precision, recall, and the AUC were fixed near 100. For the DoS-Slowloris

attack, the accuracy was 99.89, and the loss was zero, while the F1 score, precision, recall, and AUC were fixed at 99.93. On the other side, the DDOS-LOIC-UDP attack achieved an accuracy of 99.9, the loss was zero, while the F1 score, precision, recall, and AUC were fixed at 99.97. The Brute Force-Web attack achieved an accuracy of 99.5, a loss of 0.33, and fixed values of 99.54 for both the F1 score and precision, while recall and AUC were 99.55. The accuracy of the Brute Force-XSS attack was 99.91, the loss was zero, and the F1 score was 99.87, while precision, recall, and AUC were all fixed at 99.88. Last but not least, the SQL Injection attack achieved an accuracy of 99.89, a loss of zero, while the F1 score, precision, recall, and AUC were fixed at 99.99.

As described in Section 4.5, only 22 of the most important features were selected. After applying FS with the proposed HXGBLSTM, the accuracy for the FTP-BruteForce attack raised to 100, the loss was zero. According to the SSH-Bruteforce attack, the accuracy reached 99.97, while the F1 score, precision, recall, and AUC attained 99.99. Referring to DDOS attack-HOIC attack, the accuracy was 99.85, the loss was zero, and the F1 score, precision, recall, and the auc attained 99.99. For the Bot attack, the accuracy was 99.7, the loss was zero, and the F1 score, precision, recall, and AUC were fixed at 99.81. Regarding the DoS-GoldenEye attack, the results were unexpected because accuracy decreased to 97.42, the loss increased to 2.33, the F1 score was 97.5, precision attained 97.63, and recall and AUC were all set to 97.49. For the DoS-Slowloris attack, the accuracy was 99.83, and the loss was zero, while the F1 score, precision, recall, and AUC were fixed at 99.82. On the other hand, the F1 score, precision, recall, and accuracy of the DDOS-LOIC-UDP attack were all fixed at 100. The loss was also zero. The Brute Force-Web attack achieved an accuracy of 99.89, a loss of zero, while the F1 score, precision, recall, and the AUC were fixed at 99.94. The accuracy of the Brute Force-XSS attack was 99.96, the loss was zero, and the F1 score, precision, recall, and AUC were all fixed at 99.97. Last but not least, the SQL Injection attack achieved an accuracy of 99.99, a loss of zero, while the F1 score, precision, recall, and AUC were fixed at 100.

4.7.2 HXGBLSTM Vs. Other Algorithms

Similar to binary classification, we conducted in-depth comparison with the most recent DL algorithms that were employed in the cyber threat detection literature. In this comparison, we used deep learning algorithms from earlier studies such as [26], [8], BWO-CONV-LSTM [27], and [6]. The results of the comparison for deep learning approaches are shown in Table 11. Bold text in the table represents the overall best solution.

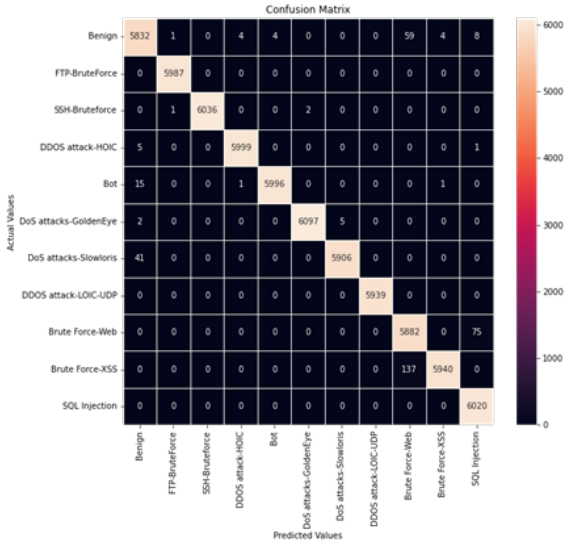
For the FTP-BruteForce attack, HXGBLSTM achieved an accuracy of 100 higher in multi-class classification than that reported in the other literature. Regarding the SSH-Bruteforce attack, our accuracy was 99.97 higher, respectively, than the other algorithms of [26], [8], BWO-CONV-LSTM [27], and [6]. For DDoS-HOIC and Bot attacks, Our accuracy was 99.89 and 99.7 respectively, greater than that of BWO-CONV-LSTM [27]. According to the DoS-GoldenEye attack, Our accuracy was 99.87 greater than the other algorithms. Regarding the DoS-Slowloris attack, Our accuracy was 99.91 higher than the other algorithms. For DDoS-HOIC and Bot attacks, Our accuracy was 99.89 and 99.7 respectively, greater than that of BWO-CONV-LSTM [27]. According to Brute Force -Web, Brute Force -XSS, and SQL Injection attacks, our accuracy reaches 99.89, 99.96, and 99.99 respectively which were the best results among other algorithms.

5 Conclusion and Future Work

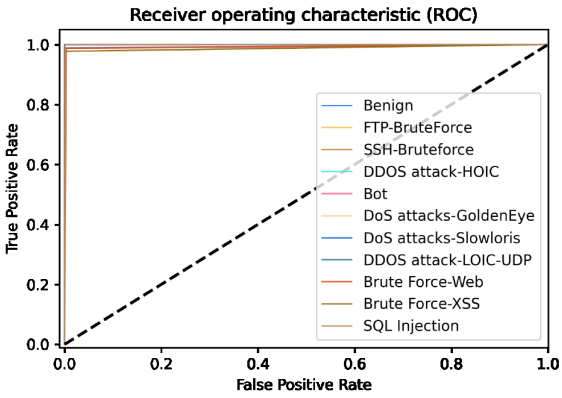
Cyber threat detection systems have been used to detect malicious and abnormal network behavior. Different kinds of these systems have been adopted to safeguard the network using a variety of DL algorithms. Optimizing system performance is the main objective of providing such consistent and effective systems. This paper proposes the HXGBLSTM algorithm to address this problem. All of the features were utilized, and feature selection algorithms were then used. To avoid selecting features that are redundant or unneeded, HXGBLSTM uses XGBoost as a feature selection algorithm. Feature selection procedures were utilized to locate the optimal features for efficient results with the least amount of complexity and processing time. The most recent CSE-CICIDS2018 dataset is used to analyze the algorithm resistance utilizing the binary or multiclass classification modes. When performance is compared, the results show that HXGBLSTM for cyber threat detection outperforms seven cutting-edge deep learning algorithms for binary classification and four for multiclass classification. In comparison to the standard LSTM, HXGBLSTM performed better. This may be because the HXGBLSTM algorithm deals with more precise features without wasting time on less significant features. In the future, it is advised to evaluate the outcomes of different feature selection algorithms and optimize features utilizing swarm intelligence algorithms. This method's applicability will be improved by testing it on various datasets with newly developed attacks.

6 Data Availability

Datasets related to this manuscript can be found at https://registry.opendata.aws/cse-cic-ids2018/, an open-source online data repository hosted at Canadian Institute for Cybersecurity [13].



(a) CM for multi-class classification before FS.

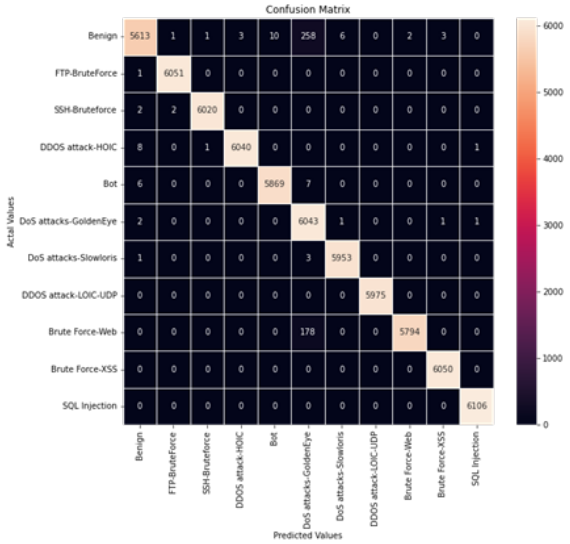


(b) ROC for multi-class classification before FS.

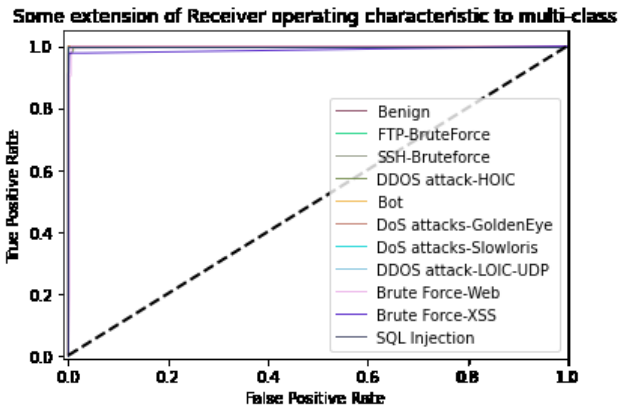
Figure 7: CM of the Standard LSTM based on multi-class classification.

Table 10: Efficiency of the proposed HXGBLSTM based on multi-class Classification.

Attack	HXGBLSTM(After FS)						Standard LSTM (Before FS)					
	Accuracy	Loss	F1	Precision	Recall	AUC	Accuracy	Loss	F1	Precision	Recall	AUC
FTP-BruteForce	**100**	**0**	**100**	**100**	**100**	**100**	99.12	0	99.5	99.52	99.49	99.72
SSH-Bruteforce	**99.97**	**0**	**99.99**	**99.99**	**99.99**	**99.99**	99.77	0	99.8	99.8	99.8	99.8
DDOS attack-HOIC	99.85	0	99.9	99.9	99.9	99.9	**99.89**	**0**	**99.95**	**99.95**	**99.95**	**99.95**
Bot	**99.7**	**0**	**99.81**	**99.81**	**99.81**	**99.81**	99.58	0.33	99.56	99.56	99.56	99.56
DoS attacks-GoldenEye	97.42	2.33	97.5	97.63	97.49	97.49	**99.87**	**0**	**100**	**100**	**100**	**100**
DoS attacks-Slowloris	**99.91**	**0**	**99.95**	**99.95**	**99.95**	**99.95**	99.89	0	99.93	99.93	99.93	99.93
DDOS attack-LOIC-UDP	**100**	**0**	**100**	**100**	**100**	**100**	99.9	0	99.97	99.97	99.97	99.97
Brute Force -Web	**99.89**	**0**	**99.94**	**99.94**	**99.94**	**99.94**	99.5	0.33	99.54	99.54	99.55	99.55
Brute Force -XSS	**99.96**	**0**	**99.97**	**99.97**	**99.97**	**99.97**	99.91	0	99.87	99.88	99.88	99.88
SQL Injection	**99.99**	**0**	**100**	**100**	**100**	**100**	99.89	0	99.99	99.99	99.99	99.99



(a) CM of HXGBLSTM based on multi-class classification.



(b) AUC-ROC of HXGBLSTM based on multi-class classification.

Figure 8: Performance of HXGBLSTM based on multi-class classification.

Table 11: Comparison with related literature based on accuracy for multi-class classification.

Attack	Proposed HXGBLSTM	[26]	[8]	[27]	[6]
FTP-BruteForce1	**100**	98	76.1	96.88	84
SSH-Bruteforce2	**99.97**	96	76.1	96.88	84
DDOS attack-HOIC3	99.89	**100**	**100**	94.67	**100**
Bot4	99.7	-	99.94	99.45	**100**
DoS attacks-GoldenEye5	**99.87**	47	99.79	99.55	96
DoS attacks-Slowloris6	**99.91**	66	99.79	99.55	96
DDOS attack-LOIC-UDP7	**100**	100	100	94.67	100
Brute Force -Web8	**99.89**	30	76.1	96.88	95
Brute Force -XSS9	**99.96**	65	76.1	96.88	95
SQL Injection10	**99.99**	8	0	98.35	95

References

[1] Al-Fawa'reh, M., Al-Fayoumi, M., Nashwan, S., and Fraihat, S. Cyber threat intelligence using pca-dnn model to detect abnormal network behavior. *Egyptian Informatics Journal 23*, 2 (2022), 173–185.

[2] Al Razib, M., Javeed, D., Khan, M. T., Alkanhel, R., and Muthanna, M. S. A. Cyber threats detection in smart environments using sdn-enabled dnn-lstm hybrid framework. *IEEE Access 10* (2022), 53015–53026.

[3] Alahmed, S., Alasad, Q., Hammood, M. M., Yuan, J.-S., and Alawad, M. Mitigation of black-box attacks on intrusion detection systems-based ml. *Computers 11*, 7 (2022), 115.

[4] Ali, A. F., and Hassanien, A.-E. A survey of metaheuristics methods for bioinformatics applications. In *Applications of Intelligent Optimization in Biology and Medicine: Current Trends and Open Problems*. Springer, 2015, pp. 23–46.

[5] Alohali, M. A., Al-Wesabi, F. N., Hilal, A. M., Goel, S., Gupta, D., and Khanna, A. Artificial intelligence enabled intrusion detection systems for cognitive cyber-physical systems in industry 4.0 environment. *Cognitive Neurodynamics* (2022), 1–13.

[6] Alzughaibi, S., and El Khediri, S. A cloud intrusion detection systems based on dnn using backpropagation and pso on the cse-cic-ids2018 dataset. *Applied Sciences 13*, 4 (2023), 2276.

[7] Ankit, T., and Ritika, L. A review of the advancement in intrusion detection datasets. *Procedia Computer Science 167* (2020), 636–645.

[8] Antunes, M., Oliveira, L., Seguro, A., Veríssimo, J., Salgado, R., and Murteira, T. Benchmarking deep learning methods for behaviour-based network intrusion detection. In *Informatics* (2022), vol. 9, p. 29.

[9] Asad, M., Asim, M., Javed, T., Beg, M. O., Mujtaba, H., and Abbas, S. Deepdetect: detection of distributed denial of service attacks using deep learning. *The Computer Journal 63*, 7 (2020), 983–994.

[10] Assis, M. V., Carvalho, L. F., Lloret, J., and Proença Jr, M. L. A gru deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications 177* (2021), 102942.

[11] Azeroual, H., Belghiti, I. D., and Berbiche, N. A framework for implementing an ml or dl model to improve intrusion detection systems (ids) in the ntma context, with an example on the dataset (cse-cic-ids2018). In *ITM Web of Conferences* (2022), vol. 46, p. 02005.

[12] Bacanin, N., Venkatachalam, K., Bezdan, T., Zivkovic, M., and Abouhawwash, M. A novel firefly algorithm approach for efficient feature selection with covid-19 dataset. *Microprocessors and Microsystems 98* (2023), 104778.

[13] Canadian Institute for Cybersecurity. A realistic cyber defense dataset (cse-cic-ids2018). https://registry.opendata.aws/cse-cic-ids2018/, June 2023.

[14] Carrio, A., Sampedro, C., Rodriguez-Ramos, A., and Campoy, P. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors 2017* (2017).

[15] Chouhan, R. K., Atulkar, M., and Nagwani, N. K. A framework to detect ddos attack in ryu controller based software defined networks using feature extraction and classification. *Applied Intelligence* (2022), 1–21.

[16] Darwish, A., Hassanien, A. E., and Das, S. A survey of swarm and evolutionary computing approaches for deep learning. *Artificial intelligence review 53*, 3 (2020), 1767–1812.

[17] Eskandari, S., and Seifaddini, M. Online and offline streaming feature selection methods with bat algorithm for redundancy analysis. *Pattern Recognition 133* (2023), 109007.

[18] Farhan, B. I., and Jasim, A. D. Performance analysis of intrusion detection for deep learning model based on cse-cic-ids2018 dataset. *Indonesian Journal of Electrical Engineering and Computer Science 26*, 2 (2022), 1165–1172.

[19] Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm. *Neural computation 12*, 10 (2000), 2451–2471.

[20] Ghimire, S., Deo, R. C., Wang, H., Al-Musaylh, M. S., Casillas-Pérez, D., and Salcedo-Sanz, S. Stacked lstm sequence-to-sequence autoencoder with feature selection for daily solar radiation prediction: a review and new modeling results. *Energies 15*, 3 (2022), 1061.

[21] Haider, S., Akhunzada, A., Mustafa, I., Patel, T. B., Fernandez, A., Choo, K.-K. R., and Iqbal, J. A deep cnn ensemble framework for efficient ddos attack detection in software defined networks. *Ieee Access 8* (2020), 53972–53983.

[22] Hind, B., and Barbora, B. Recent advances in machine-learning driven intrusion detection in transportation: Survey. *Procedia Computer Science 184* (2021), 877–886.

[23] Hnamte, V., and Hussain, J. Dependable intrusion detection system using deep convolutional

neural network: A novel framework and performance evaluation approach. *Telematics and Informatics Reports 11* (2023), 100077.

[24] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[25] Javeed, D., Gao, T., Khan, M. T., and Shoukat, D. A hybrid intelligent framework to combat sophisticated threats in secure industries. *Sensors 22*, 4 (2022), 1582.

[26] Jiyeon, K., Yulim, S., and Eunjung, C. An intrusion detection model based on a convolutional neural network. *Journal of Multimedia Information System 6*, 4 (2019), 165–172.

[27] Kanna, P. R., and Santhi, P. Hybrid intrusion detection using mapreduce based black widow optimized convolutional long short-term memory neural networks. *Expert Systems with Applications 194* (2022), 116545.

[28] Kilincer, I. F., Ertam, F., and Sengur, A. A comprehensive intrusion detection framework using boosting algorithms. *Computers and Electrical Engineering 100* (2022), 107869.

[29] Malik, J., Akhunzada, A., Bibi, I., Imran, M., Musaddiq, A., and Kim, S. W. Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in sdn. *IEEE Access 8* (2020), 134695–134706.

[30] Mijwil, M., Salem, I. E., and Ismaeel, M. M. The significance of machine learning and deep learning techniques in cybersecurity: A comprehensive review. *Iraqi Journal For Computer Science and Mathematics 4*, 1 (2023), 87–101.

[31] Mishra, S., Sagban, R., Yakoob, A., and Gandhi, N. Swarm intelligence in anomaly detection systems: an overview. *International Journal of Computers and Applications 43*, 2 (2021), 109–118.

[32] Mohamed, A. F., Leandros, M., Sotiris, M., and Helge, J. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications 50* (2020), 102419.

[33] Mossa, G., Ghaleb, G., Faisal, A., Reem, A., and Suad, O. A detailed analysis of benchmark datasets for network intrusion detection system. *Asian Journal of Research in Computer Science 7*, 4 (2021), 14–33.

[34] Ragab, M. Hybrid firefly particle swarm optimisation algorithm for feature selection problems. *Expert Systems* (2023).

[35] Ren, X., Yang, W., Jiang, X., Jin, G., and Yu, Y. A deep learning framework for multimodal course recommendation based on lstm+ attention. *Sustainability 14*, 5 (2022), 2907.

[36] Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE 109*, 3 (2021), 247–278.

[37] Sohail, A. Genetic algorithms in the fields of artificial intelligence and data sciences. *Annals of Data Science 10*, 4 (2023), 1007–1018.

[38] Thakkar, A., and Lohiya, R. Role of swarm and evolutionary algorithms for intrusion detection system: A survey. *Swarm and evolutionary computation 53* (2020), 100631.

[39] Thuy, T. T. T., Thuan, L. D., Duc, N. H., and Minh, H. T. A study on heuristic algorithms combined with lr on a dnn-based ids model to detect iot attacks. In *MENDEL* (2023), vol. 29, pp. 62–70.

[40] Tiwari, A. A hybrid feature selection method using an improved binary butterfly optimization algorithm and adaptive β–hill climbing. *IEEE Access* (2023).

[41] Trojovská, E., Dehghani, M., and Trojovskỳ, P. Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm. *IEEE Access 10* (2022), 49445–49473.

[42] Ullah, S., Khan, M. A., Ahmad, J., Jamal, S. S., e Huma, Z., Hassan, M. T., Pitropakis, N., and Buchanan, W. J. Hdl-ids: a hybrid deep learning architecture for intrusion detection in the internet of vehicles. *Sensors 22*, 4 (2022), 1340.

[43] Vyshnia, G. Feature importance with featurewiz python. https://www.kaggle.com/code/gvyshnya/jan22-tpc-feature-importance-with-featurewiz#Introduction, Jan 2022.

[44] Wang, F., Zhang, W., Yang, Q., Kang, Y., Fan, Y., Wei, J., Liu, Z., Dai, S., Li, H., Li, Z., et al. Generation of a hutchinson–gilford progeria syndrome monkey model by base editing. *Protein & cell 11*, 11 (2020), 809–824.

[45] Wang, Y.-C., Houng, Y.-C., Chen, H.-X., and Tseng, S.-M. Network anomaly intrusion detection based on deep learning approach. *Sensors 23*, 4 (2023), 2171.

[46] Yuan, S., and Wu, X. Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security 104* (2021), 102221.

[47] Zhang, B., Zhang, Y., and Jiang, X. Feature selection for global tropospheric ozone prediction based on the bo-xgboost-rfe algorithm. *Scientific Reports 12*, 1 (2022), 1–10.

[48] Zhang, Y., and Liu, Q. On iot intrusion detection based on data augmentation for enhancing learning on unbalanced samples. *Future Generation Computer Systems 133* (2022), 213–227.

[49] Zhou, C., and Chen, X. Predicting china's energy consumption: Combining machine learning with three-layer decomposition approach. *Energy Reports 7* (2021), 5086–5099.

[50] Zhou, L., Zhang, C., Liu, F., Qiu, Z., and He, Y. Application of deep learning in food: a review. *Comprehensive reviews in food science and food safety 18*, 6 (2019), 1793–1811.