Scientific Research Publishing

Open Journal of Applied Sciences, 2024, 14, 2296-2319
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

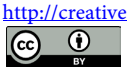
Intelligent Detection and Identification of Attacks in IoT Networks Based on the Combination of DNN and LSTM Methods with a Set of Classifiers

Brou Médard Kouassi, Vincent Monsan, Kablan Jérôme Adou

Department of Mathematics and Computer Science, University Felix Houphouet-Boigny, Abidjan, Côte d'Ivoire
Email: medardkoisy@gmail.com

How to cite this paper: Kouassi, B.M., Monsan, V. and Adou, K.J. (2024) Intelligent Detection and Identification of Attacks in IoT Networks Based on the Combination of DNN and LSTM Methods with a Set of Classifiers. *Open Journal of Applied Sciences*, **14**, 2296-2319.
https://doi.org/10.4236/ojapps.2024.148153

Received: July 21, 2024
Accepted: August 24, 2024
Published: August 27, 2024



Abstract

Internet of Things (IoT) networks present unique cybersecurity challenges due to their distributed and heterogeneous nature. Our study explores the effectiveness of two types of deep learning models, long-term memory neural networks (LSTMs) and deep neural networks (DNNs), for detecting attacks in IoT networks. We evaluated the performance of six hybrid models combining LSTM or DNN feature extractors with classifiers such as Random Forest, k-Nearest Neighbors and XGBoost. The LSTM-RF and LSTM-XGBoost models showed lower accuracy variability in the face of different types of attack, indicating greater robustness. The LSTM-RF and LSTM-XGBoost models show variability in results, with accuracies between 58% and 99% for attack types, while LSTM-KNN has higher but more variable accuracies, between 72% and 99%. The DNN-RF and DNN-XGBoost models show lower variability in their results, with accuracies between 59% and 99%, while DNN-KNN has higher but more variable accuracies, between 71% and 99%. LSTM-based models are proving to be more effective for detecting attacks in IoT networks, particularly for sophisticated attacks. However, the final choice of model depends on the constraints of the application, taking into account a trade-off between accuracy and complexity.

Keywords

Internet of Things, Machine Learning, Attack Detection, Jamming, Deep Learning

1. Introduction

Radiocommunication, the technique of transmitting information over long dis-

tances using electromagnetic waves, offers undeniable advantages for modern society. Its impact is palpable in all areas of life, and strongly influences human development. Among its many applications, which continue to evolve at an exponential rate over time and generate large volumes of sensitive and useful data for humanity, are connected objects [1]. The Internet of Things, which connects billions of networked devices and sensors of all kinds [2] [3], has been growing by leaps and bounds in recent years. This development offers a multitude of opportunities, and considerably increases the risk of network and communicating object disruptions due to cyber-attacks. Indeed, IoT devices, often with limited resources and inadequate security measures, are prime targets for hackers [4]. Detecting and identifying attacks on IoT networks therefore represents a major challenge for the security of critical infrastructures and the protection of sensitive data [5]. Traditional intrusion detection methods, based on predefined rules, often prove ineffective in the face of increasingly diverse and evolving threats [6] [7]. Machine Learning offers a promising solution to this challenge. By analyzing large volumes of data from IoT devices, Machine Learning algorithms can learn to identify disruptions or abnormal behavior in IoTs and detect attacks in real time.

This study presents an innovative ensemble model for detecting and identifying disruptions via attacks in IoT networks. The proposed model combines powerful machine learning algorithms to overcome the limitations of traditional, individual approaches and improve detection accuracy. The proposed ensemble model consists of two layers. A feature extraction layer that uses artificial neural networks and long-term memory neural networks to extract relevant features from IoT network data. And a second classification layer that employs Random Forest, K-Nearest Neighbors and Extreme Gradient Boosting algorithms to classify the extracted data and identify potential attacks. The combination of algorithms significantly reduces model learning time. The model adapts efficiently to small datasets, a common challenge in IoT environments. Exploiting the strengths of each algorithm results in more accurate and reliable attack detection.

This research makes a significant contribution to the field of IoT network security by introducing a novel ensemble model. The proposed architecture, which combines deep learning (ANN and LSTM) with traditional machine learning algorithms (RF, KNN, XGBoost), makes it possible to extract discriminating information from data and considerably improve the accuracy of attack detection. What's more, by being specifically designed to handle the limited data constraints of IoT environments, this model offers a robust and effective solution

This work is organized as follows. After the state of the art is presented in the first section, the material and method are presented in Section 2, while the experimental setup is presented in Section 3. Section 4 shows the experimental results, while Section 5 presents the discussion and Section 6 the conclusion of the paper.

2. Related Work

For wireless sensor networks jamming is the most feared security threat. Indeed, it can easily neutralize even wireless sensor networks equipped with solid upper-layer security mechanisms. The seriousness of jamming lies in its ability to go undetected, bypassing these protections.

Many researchers have carried out studies in this area, including Arjoune *et al.* who carried out an extensive study of the performance of machine learning models in the context of jamming attack detection. They used a large-scale dataset to train, validate and test three popular algorithms: random forest, support vector machine and neural network [8]. Their experiments show that the random forest technique outperforms the other approaches in terms of accuracy and efficiency. It achieves a detection rate (Pd) of 97.5%, compared with 96.4% and 97.1% for the neural network and cubic support vector machine respectively. The study does not take into account the diversity of attack types and their impact on model performance.

Hachimi. al have developed a security system for 5G networks [9]. This system uses artificial intelligence to recognize different types of attack, such as jamming, with a success rate of 94.51%.

Jamming attacks aimed at hindering communication capabilities are becoming a critical aspect of wireless networks. The detection of reactive jammers that detect the spectrum and attack the network only when a legitimate communication is in progress. With this in mind, Arcangeloni *et al.* have developed a method for detecting interference in wireless networks [10]. Using sensors placed around the network, they can identify signals that disrupt communications. Their tests showed that this method is reliable, with a detection rate of 90% and few false alarms.

Yakub Kayode Saheed *et al.* proposed an intrusion detection system based on six machine learning models (ML-IDS) to detect attacks on IoT networks [11]. Feature scaling was performed using the minimum-maximum (min-max) normalization concept on the UNSW-NB15 dataset to limit information leakage on test data. Dimensionality reduction was then performed using principal component analysis (PCA) [12]. The experimental results of our findings were evaluated in terms of validation data set, precision, area under the curve, recall, F1, precision, kappa and Mathew correlation coefficient (MCC). Results were also compared with existing work, and our results were competitive with a precision of 99.9% and an MCC of 99.97%.

Chucher *et al.* have studied the use of various machine learning methods for intrusion detection in computer systems [13]. Among the algorithms studied were nearest neighbors (KNN), support vector machine (SVM), decision tree (DT), naive Bayes (NB), Random Forest (RF), artificial neural network (ANN) and logistic regression (LR). The performance of these algorithms was compared for binary and multi-class classification on a dataset called Bot-IoT. Several key parameters were evaluated, including precision, accuracy, recall, F1 score and

log loss. The results showed that for the detection of HTTP Distributed Denial of Service (DDoS) attacks, the Random Forest (RF) algorithm achieves a remarkable 99% accuracy. Furthermore, simulations based on the aforementioned evaluation parameters revealed that RF outperforms other ML algorithms for all attack types in the context of binary classification.

Alissa *et al.* developed a binary classification method based on machine learning [14]. They used a specific dataset (UNSW-NB15) and applied a technique to balance the data (SMOTE). After a thorough analysis of the data, they chose to use a decision tree algorithm. The results show that this method performs very well, with an accuracy of 94%.

Qiang Xu *et al.* have proposed a new method for detecting intrusions into networks [15]. They tested several algorithms and concluded that the "Naïve Bayes" algorithm was the best for distinguishing normal activities from attacks.

Pirayesh *et al.* conducted an extensive study on jamming attacks in wireless networks [16]. They have classified and analysed different types of attacks, showing their impact on various networks such as Wi-Fi, Bluetooth and cellular networks. Their work provides a comprehensive reference on jamming techniques and possible countermeasures.

3. Materials and Methods

3.1. Materials

3.1.1. Database Description

UNSW-NB15 is a network traffic dataset widely used for research and development in the field of cybersecurity. It provides a valuable resource for training and evaluating intrusion detection and network traffic analysis models. The "UNSW-NB15" dataset was created by the UNSW Canberra research team in collaboration with the Australian Cyber Security Centre (ACCS). The ACCS Cyber Range Lab's Perfect Storm tool was used to generate realistic synthetic attack behaviors, which were then integrated with a real network traffic dataset collected from various network environments. The data set includes a wide range of real-life network activities, from Web traffic and file transfers to instant messaging and online gaming.

In addition to real network activity, UNSW-NB15 also includes synthetic attack behaviors generated using the Perfect Storm tool. These synthetic attacks represent realistic, modern threats, enabling intrusion detection models to be trained on a wide range of attack scenarios. Network packets in the UNSW-NB15 dataset are carefully categorized according to their type of activity (normal or attack) and the specific type of attack (e.g. DDoS, port scan, SQL injection). This categorization facilitates performance evaluation of intrusion detection models.

Table 1 shows the breakdown of attacks by class and frequency.

The frequency distribution of the labels is as follows: 93,000 for the normal label and 157,982 for the attacks. Figure 1 gives a graphical representation of the attack class distribution.

Table 1. Breakdown of attacks by class and frequency.

Attacks by class (at-tack_cat)	Distribution of categories in the training set (80%)	Breakdown of categories in the test set (20%)	Total frequency measurements
Normal	*74,400*	*18,600*	*93,000*
Generic	47,097	11,774	58,871
Exploits	35,620	8905	44,525
Fuzzers	19,397	4849	24,246
DoS	13,082	3271	16,353
Reconnaissance	11,189	2798	13,987

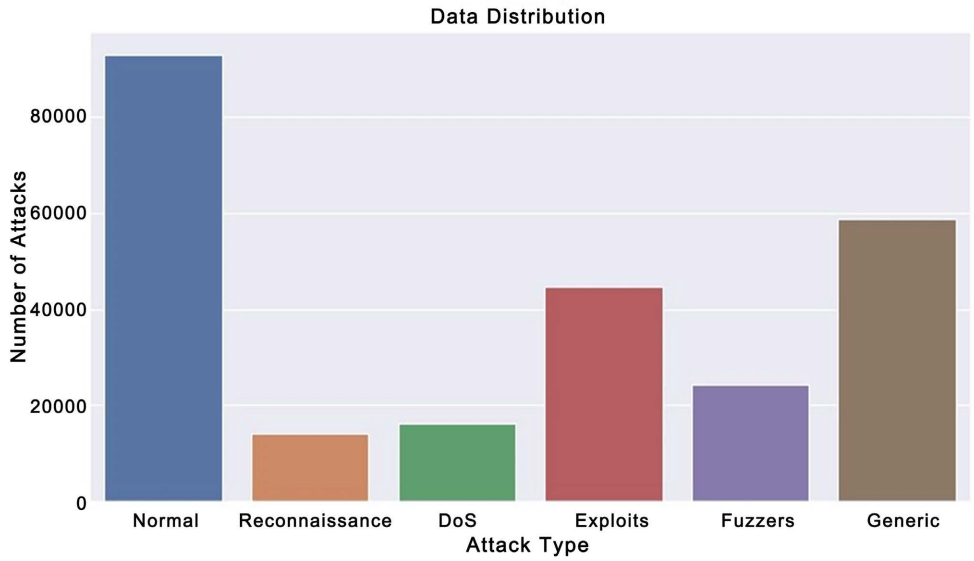


Figure 1. Graphic representation of attack class distribution.

3.1.2. Material for Setting up the Experiment

Our model experiments were trained on a computer with a Windows 10 configuration featuring an Intel(R) Core™ i7-8650U processor, 16 GB RAM and an NVIDIA GeForce graphics card. The models were developed in Python using the Keras version 2.4 API with the Tensorflow version 2.4 backend and CUDA/CuDNN dependencies to optimize performance thanks to GPU acceleration.

3.2. Algorithms Studied

3.2.1. Deep Neural Networks (DNN)

Deep Neural Networks (DNNs) are a type of machine learning algorithm inspired by the workings of the human brain [17] [18]. They are composed of several layers of interconnected artificial neurons, which enable DNNs to learn and process complex information. In its operation, each artificial neuron uses a simple computing unit that receives inputs, processes them and produces an output. Neurons are connected to each other by weights, which represent the strength of the connection between neurons [19]. DNNs can learn from large amounts of

data and identify complex patterns. DNNs can be adapted to a wide variety of tasks and problems. They are powerful tools that have revolutionized artificial intelligence and have the potential to solve many complex problems in a variety of fields.

In short, DNNs represent a revolutionary technology within artificial intelligence, offering remarkable learning and adaptation capabilities for solving complex problems in a multitude of fields.

3.2.2. LSTM

Long-term memory networks (LSTMs) are a type of recurrent neural network (RNN) architecture specially designed to solve the problem of gradient vanishing, which is a common problem in traditional RNNs [20]. LSTM networks are capable of learning long-term dependencies in data, making them well suited to tasks such as speech recognition, natural language processing and time series prediction. An LSTM network consists of a cell state, an input gate, an output gate and a forgetting gate [21]. These gates regulate the flow of information into and out of the cellular state, enabling the network to selectively remember or forget information over long periods.

The cell state, the central memory unit of an LSTM network, stores information that can be accessed and updated over long periods. The gateway controls the flow of new information into the cell state. It determines which new information is relevant to the current task and should be added to the cell state. The output gate controls the flow of information between the cell state and the output of the LSTM unit. It determines which cell state information is relevant to the current network output. The forget gate controls the flow of information out of the cell state. It determines which cell state information is no longer relevant and should be forgotten.

3.2.3. XGBoost

XGBoost, which stands for eXtreme Gradient Boosting, is a powerful machine learning algorithm belonging to the category of ensemble learning methods, in particular gradient boosting. It is known for its efficiency in various tasks such as classification: Prediction of a discrete category and regression: Prediction of a continuous value [22]. XGBoost uses decision trees as basic learners. These are relatively simple models that divide data according to characteristics (attributes) to make predictions at the basic learning level.

In sequential learning, unlike traditional decision trees, XGBoost builds these trees sequentially. Each subsequent tree focuses on correcting errors made by previous trees. This sequential approach improves the overall accuracy of the model.

For gradient boosting, XGBoost uses a technique called "gradient boosting". At each iteration, it calculates the "gradient", which essentially represents the errors made by the current model. The following tree is then constructed to specifically correct these errors, resulting in a more accurate model.

XGBoost also incorporates regularization techniques to avoid over-fitting. This allows the model to better adapt to new data. Over-fitting occurs when a model becomes too specific to training data and performs poorly on new data.

It's an algorithm known for its excellent accuracy in various machine learning tasks. It can efficiently process large datasets thanks to its parallel processing capabilities. The feature importance analysis provided by XGBoost helps to understand which features have the most significant impact on model predictions. XGBoost can efficiently handle missing values, making it suitable for real-world datasets that often contain such data.

To further optimize the objective function, XGBoost uses a second-order Taylor expansion to approximate the objective function, and the optimal solution is the quadratic optimal solution [23]. In addition, a regular term is added to regulate the complexity of the spanning tree, reducing the risk of mode overfitting. The loss function described is as follows

$$f_{obj}^{(t)} = \sum_{n}^{i}\left[L\left(y_i, \hat{y}_l^{t-1}\right) + f_t\left(x_i\right) + \frac{1}{2}L^n \right]\left(y_i, \hat{y}_l^{t-1}\right) + f_t^2\left(x_i\right) + \Omega\left(f_t\right) \quad (1)$$

$$\Omega\left(f_t\right) = \frac{1}{2}\lambda\sum_{j}^{T}\left\|w_j\right\|_2 + \gamma T \quad (2)$$

In this equation, w_j represents the weights associated with the tree's leaf nodes. T corresponds to the total number of nodes present in the tree. The hyperparameters λ and γ play a crucial role in controlling node complexity.

The XGBoost technique uses the narrowing strategy [24] to cluster weak learners and reduce the probability of model overfitting.

This set takes the following form

$$F_m = F_{m-1}\left(X\right) + \eta f_m\left(X\right) \text{ avec } 0 < \eta < 1 \quad (3)$$

$f_m\left(X\right)$ represents the iteration function used to generate the weak learner at the same iteration. On the other hand, $F_m\left(X\right)$ denotes the iteration function employed to generate the integrated learner at the same iteration.

The parameter η shows a strong negative correlation with the number of iterations. As a result, the model tends to exhibit better generalization properties when the value of η is smaller. This is because a smaller η slows down the learning process and limits the risk of overfitting.

Overall, XGBoost is a powerful and versatile machine learning algorithm that can be a valuable tool for various data science tasks for its accuracy, scalability and interpretability.

3.2.4. Random Forest

The random forest stands out in the field of machine learning as a robust and versatile ensemble learning algorithm [25]. Renowned for its accuracy and reliability in classification and regression tasks, random forest has gained wide recognition in various fields.

At the heart of the random forest is the concept of combining multiple decision trees into a single predictive model [26]. These decision trees are con-

structed randomly, which introduces an element of chance at two levels.

When building each tree, a random subset of features is chosen from the available set. This approach reduces the model's dependence on a particular feature, thus mitigating the risk of over-fitting.

At each node of the tree, a split point is randomly selected from the possible values of the chosen feature. This further diversifies the trees, preventing them from becoming too similar.

Unlike a single decision tree, which can succumb to over-fitting, the Random Forest aggregates the predictions of all trees within the ensemble. For classification tasks, the final prediction is determined by the majority vote among the trees. In regression tasks, the average of individual predictions is used.

The random nature of the Random Forest and its ensemble approach give it several advantages [27]. Random Forest consistently delivers exceptional accuracy on complex classification and regression problems. The injection of randomness and the aggregation of multiple trees effectively reduce the model's sensitivity to over-fitting, ensuring its generalizability to unseen data. The Random Forest provides information on the relative importance of features for prediction, helping to understand the data. Random Forest learning can be efficiently parallelized, making it suitable for processing large datasets.

Random Forest's versatility and efficiency have led to its adoption in a wide range of applications. In this study, we have chosen to take a different approach to analyzing intrusion behavior by employing the Random Forest (RF) algorithm. This algorithm differs from other approaches in its ability to construct and combine multiple decision trees, resulting in more accurate and robust predictions.

The loss function is generated in terms of the following objective function F:

$$F = \sum_{x=1}^{n} l\left(A_x, \hat{A}_x\right) + \left(\gamma T - \left(1-\gamma\right)T\right) + \frac{1}{2}\lambda \sum_{y=1}^{T} w_y^2 \tag{4}$$

where n represents the features given with the label instances and $\sum_{x=1}^{n} l\left(A_x, \hat{A}_x\right)$ denotes the learning loss function, which adjusts the data in the leaf node norm L according to the following equation:

$$L = \left(\gamma T - \left(1-\gamma\right)T\right) + \frac{1}{2}\lambda \sum_{y=1}^{t} w_y^2 \tag{5}$$

The Random Forest is a testament to the power of ensemble learning, combining multiple decision trees to achieve remarkable accuracy and robustness. Its ability to handle missing data, provide information on feature importance and train efficiently on large datasets further enhances its appeal. As a result, the Random Forest has become an indispensable tool in the arsenal of machine learning practitioners, finding applications in diverse fields across different industries.

3.2.5. KNN

The K-nearest neighbor (KNN) algorithm is a simple and effective supervised

machine learning algorithm used for classification and regression [28]. It works by classifying a new data point according to the majority of classes of its K nearest neighbors in feature space.

The KNN algorithm requires a training dataset composed of labeled data points. Each data point is represented by a feature vector and a class label. The key parameter of the KNN algorithm is the value of K, which represents the number of nearest neighbors to be considered for classification or regression [29]. The value of K is generally chosen according to experience and the nature of the data. For a new data point to be classified or predicted, the algorithm calculates the distance between this point and each of the training data points. The distance used can be the Euclidean distance, the Manhattan distance or other similarity measures.

The K training data points closest to the new data point are identified on the basis of the distances calculated. KNN uses an approach based on distance and majority voting to determine the class or predicted value of a new data point [30]. The distance between two data points in feature space is usually calculated using the Euclidean distance, defined by the following formula:

$$\text{distance}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{6}$$

where

distance(x, y) represents the distance between points with coordinates x and y.

d represents the dimensionality of the data (number of features).

x_i and y_i represent the values of the ith feature for data points x and y respectively.

For classification, the new data point is assigned the most frequent class among the K nearest neighbors.

Whereas for regression, the mean or median value of the K nearest neighbors is assigned to the new data point.

The KNN algorithm is a simple, robust and versatile machine learning algorithm that can be used for a variety of classification and regression tasks. It is particularly useful for small and medium-sized datasets and when it is difficult to build a complex classification model. However, it is important to choose the K value appropriately and to consider the dimensionality of the data to obtain optimum performance.

3.3. Methods

Our methodology is based on a rigorous, step-by-step process, ensuring a structured and reliable approach to achieving our objectives, and is divided into four main stages: data pre-processing, feature extraction, model training and model validation.

- Data pre-processing: In this step, we examined the distribution of the data and noticed that some attack categories were under-represented, with less than 10,000 samples. To avoid any bias in our analyses, we chose to remove

these under-represented categories. The new dataset thus created served as the basis for the rest of our study.

- Feature extraction: To extract relevant features from the data, we tested two deep learning-based approaches: classical neural networks (DNNs) and recurrent neural networks of the LSTM type. Both types of architecture demonstrated their ability to efficiently capture the complex patterns present in attack detection data.
- Model training: In the training phase, we divided the dataset into two parts: 20% for test data and 80% for training and validation. We used 10-fold cross-validation to optimize model hyperparameters. Three classifiers were evaluated: Random Forest, k-nearest neighbors (KNN) and gradient boosting (XGBoost).
- Model validation: Finally, once the models had been trained, we tested them on the test dataset reserved for this stage. This enabled us to assess their actual performance in terms of detecting different types of attack, thus validating the robustness and generalizability of our approaches.

This rigorous methodology, combining pre-processing, feature extraction, training and validation, enabled us to obtain reliable results and identify the best performing models for attack detection in our specific context. Figure 2 describes all the stages in our methodology.

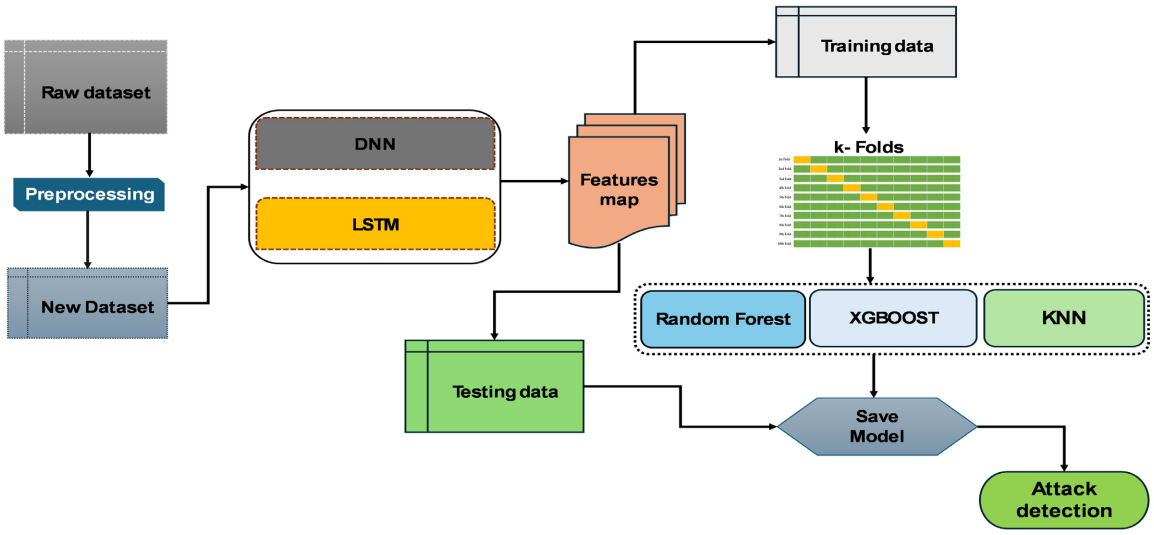


Figure 2. Illustration of our methodology.

3.4. Evaluation Metrics

The effectiveness of the model developed in this study will be evaluated using several key performance indicators. These indicators will quantify the model's ability to correctly identify data and deliver accurate results.

Accuracy: accuracy measures the proportion of positive cases correctly identified by the model. It is calculated as the ratio between the number of true positives (TP) and the sum of true positives (TP) and false positives (FP).

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \tag{7}$$

Recall: recall measures the proportion of true positive cases that the model has succeeded in identifying. It is calculated as the ratio between the number of true positives (TP) and the sum of true positives (TP) and false negatives (FN).

$$\text{Rappel} = \text{TP}/(\text{TP} + \text{FN}) \tag{8}$$

F1 score: the F1 score is a balanced measure that combines precision and recall. It is calculated as the harmonic mean of precision and recall.

$$\text{F1 Score} = 2 * (\text{Precision} * \text{Rappel})/(\text{Precision} + \text{Rappel}) \tag{9}$$

Matthews Correlation Coefficient (MCC): The Matthews Correlation Coefficient (MCC) is a more robust measure than precision and recall, as it takes into account false positives and false negatives. It ranges from −1 to +1, where +1 indicates perfect classification and −1 a completely erroneous classification.

$$\text{MCC} = \frac{\text{TP} * \text{TN} - \text{FP} * \text{FN}}{\sqrt{((\text{TP} + \text{FP}) * (\text{TP} + \text{FN}) * (\text{TN} + \text{FP}) * (\text{TN} + \text{FN}))}} \tag{10}$$

Root mean square error (RMSE): The root mean square error (RMSE) measures the mean deviation between the values predicted by the model and the actual values. It is calculated as the average of the squared differences between predicted and actual values.

$$\text{EQM} = \frac{1}{N} * \sum (y_i - \hat{y}_i)^2 \tag{11}$$

ROC-AUC curve: the ROC (Receiver Operating Characteristic) curve, also known as the ROC-AUC curve, is a powerful evaluation tool for assessing a model's ability to correctly distinguish classes in a classification problem.

The ROC curve is a graphical representation that illustrates the performance of a classification model at different decision thresholds. It consists of two axes: the y-axis represents the true positive rate (TPR), while the x-axis represents the false positive rate (FPR). The TPR indicates the proportion of positive cases correctly identified by the model, while the FPR indicates the proportion of negative cases incorrectly classified as positive.

The area under the ROC curve (AUC) is a quantitative measure of the classification model's performance. It represents the probability that the model will correctly classify a randomly selected item from the positive and negative classes. A high AUC (close to 1) indicates excellent model discrimination, while a low AUC (close to 0) suggests poor performance.

4. Results

Our results will be presented in two stages: first the results based on DNN extractors, then those based on LSTM. These extractors will be coupled with Random Forest, KNN and XGBOOST classifiers.

Case 1: DNN

Table 2 shows the results of three classification models (DNN-RF, DNN-KNN, DNN-XGBoost) for five types of attack (Normal, Reconnaissance, DoS, Exploits, Fuzzers, Generic). Each cell in the table represents the accuracy (in %) of the model for a given attack type. The DNN-RF, DNN-KNN and DNN-XGBoost models show an average accuracy of 85.5%, 83.5% and 84.5% respectively. The DNN-RF and DNN-XGBoost models show lower variability in their results, with accuracies between 59% and 99%, while DNN-KNN has higher but more variable accuracies, between 71% and 99%. The models perform differently for each type of attack:

- Normal: All models have high accuracy ranging from 91% to 92%.
- Reconnaissance: The models have an identical accuracy of 84%.
- DoS: DNN-KNN and DNN-XGBoost perform slightly better, ranging from 55% to 65%.
- Exploits: DNN-XGBOOST has the best performance at 87%, followed by DNN-RF at 85% and DNN-KNN (81%).
- Fuzzers: Performance ranges from 71% to 74%.
- Generic: All models are 99% accurate.

Table 2. DNN-based attack type performance.

Attack Type	DNN-RF	DNN-KNN	DNN-XGBoost
Normal	***93***	*91*	*92*
Reconnaissance	84	84	84
DoS	59	65	55
Exploits	85	81	87
Fuzzers	74	71	72
Generic	99	99	99

Table 3. Performance of DNN-based classification models.

Models	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	MCC (%)	MSE
DNN-RF	81.32	80.51	81.32	80.56	75.31	1.21
DNN-KNN	79.09	78.73	79.09	78.84	72.37	1.42
DNN-XGBoost	80.88	79.79	80.88	79.41	74.85	1.29

Table 3 shows the results of three classification models (DNN-RF, DNN-KNN, DNN-XGBoost) evaluated on several performance metrics: precision, recall, F1-score, MCC (Matthews Correlation Coefficient) and MSE (Mean Squared Error). Each row of the table represents the values of these metrics for a given model.

The DNN-RF model achieved the best overall performance, with the best values for precision, recall, F1-score, MCC and MSE. DNN-XGBoost performed slightly worse, while DNN-KNN had the lowest performance of the three mod-

els. However, the differences in performance between the models are relatively small.

Figure 3 shows the confusion matrix for the DNN-RF classification models. The confusion matrix shown in the image is a tool used to evaluate the performance of a supervised classification model. It indicates the number of correct and incorrect predictions made by the DNN-RF model for each class.

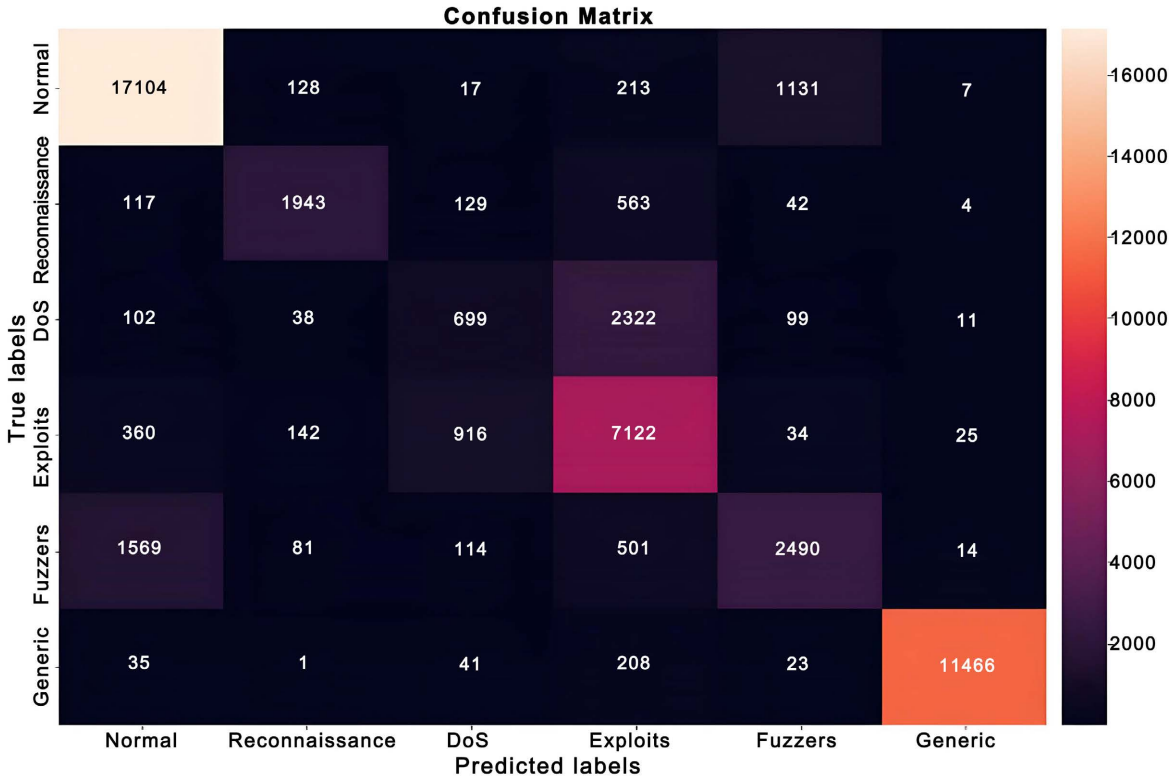


Figure 3. Confusion Matrix for DNN-RF.

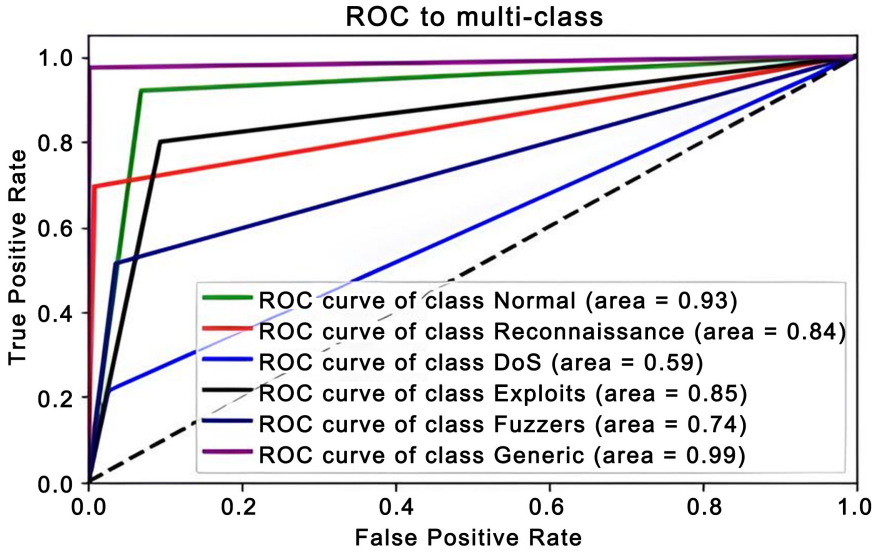


Figure 4. ROC Curves of class for DNN-RF.

Figure 4 shows the receiver operating characteristic (ROC) curve, a graphical tool used to evaluate the performance of a binary classification model. It shows the relationship between the true positive rate (TPR) and the false positive rate (FPR) of the model at different classification thresholds for DNN-RF.

Figure 5 shows a confusion matrix for the DNN-KNN classification models. This grid-like table, a common tool in supervised learning, shows the performance of the model for each class. It details the number of correct classifications and misclassifications made by the DNN-KNN model.

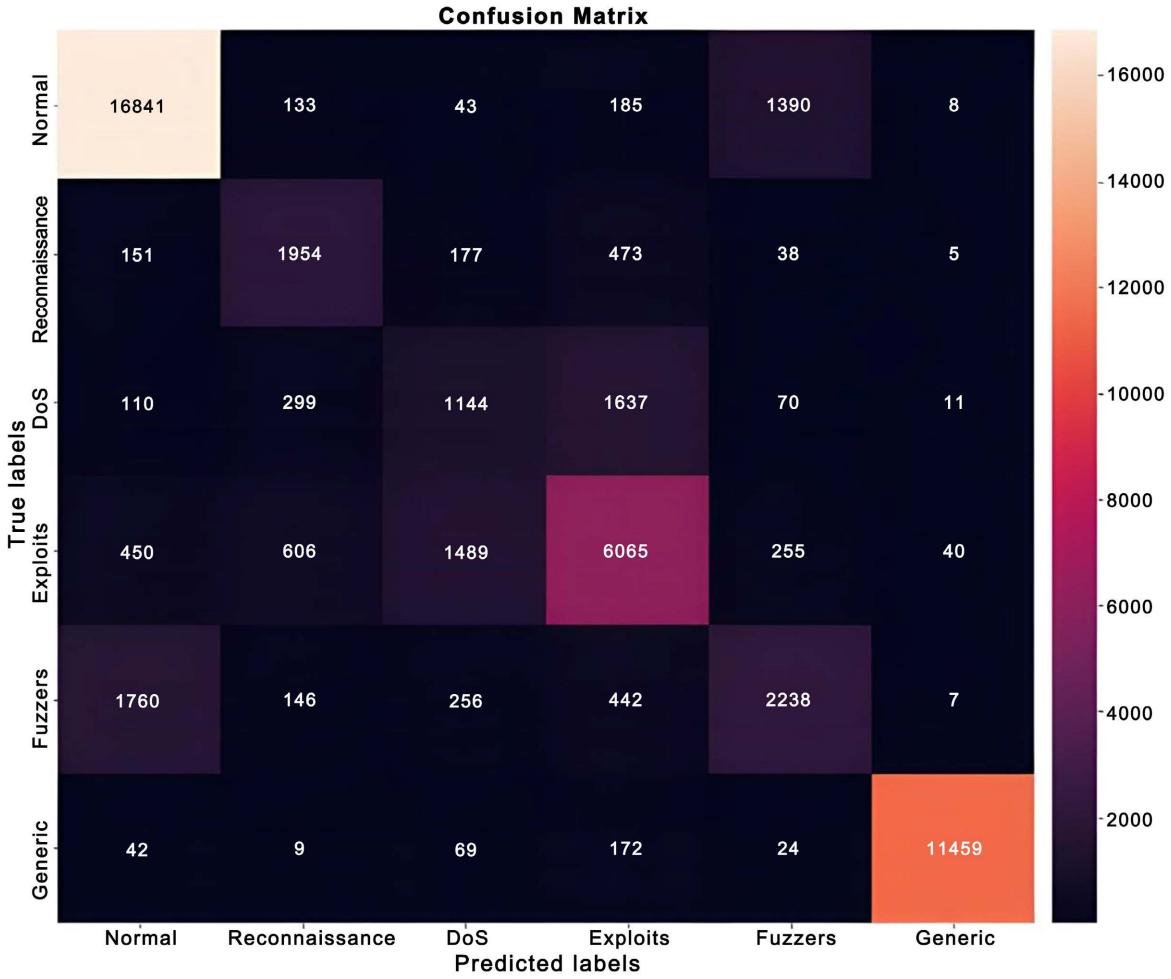


Figure 5. Confusion Matrix for DNN-KNN.

The performance of the DNN-RF model is visualized using a Receiver Operating Characteristic (ROC) curve in Figure 6. This graphical tool is commonly used for binary classification models. It depicts the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at various classification thresholds for the DNN-KNN model.

Figure 7 presents a confusion matrix for the DNN-XGBoost classification model. This table is a valuable tool for evaluating the performance of supervised classification models. It details the number of correct and incorrect predictions

made by the DNN-XGBoost model, categorized by their actual class.

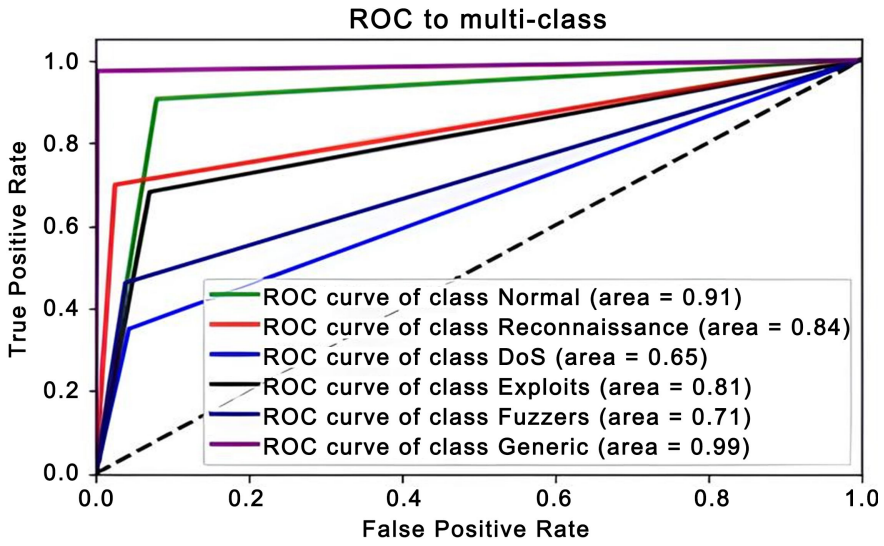


Figure 6. Roc Curves of class for DNN-KNN.

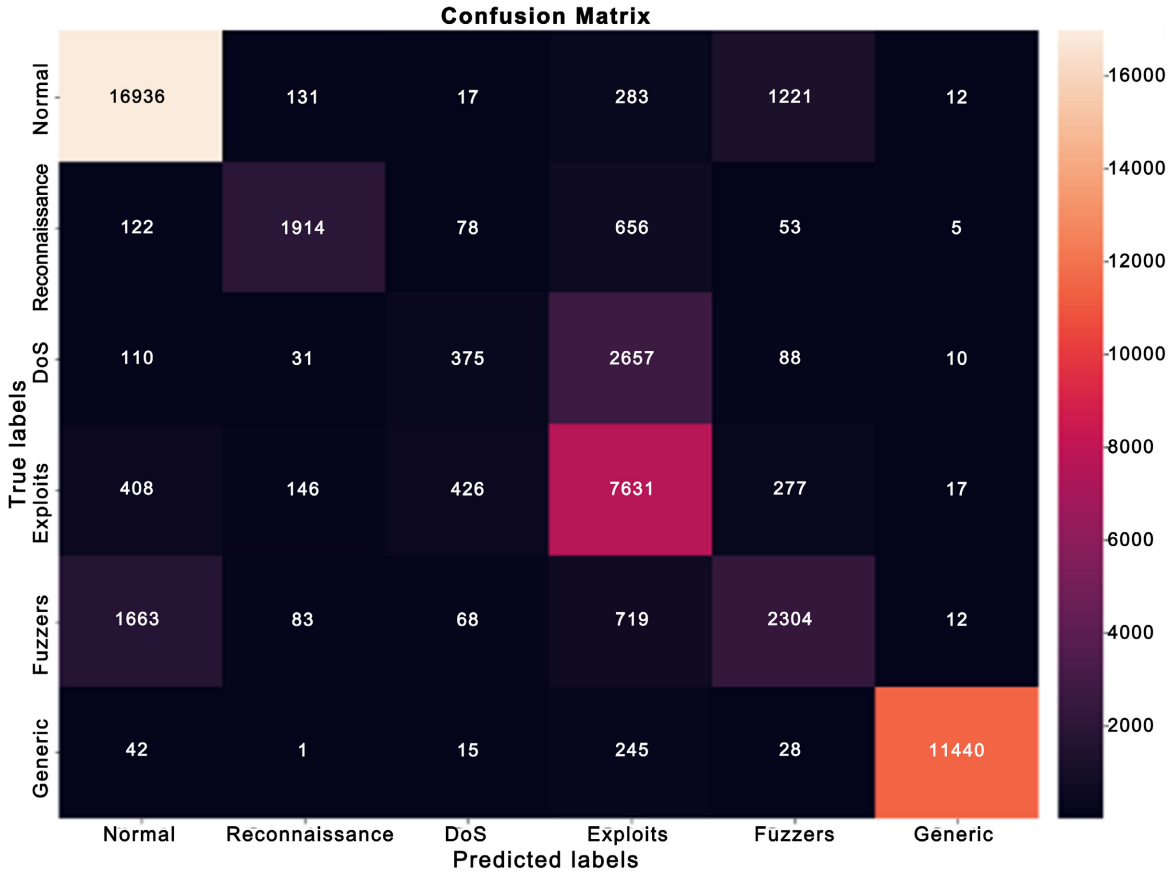


Figure 7. Confusion Matrix for DNN-XGBoost.

Figure 8 depicts the ROC curve for the DNN-XGBoost model. This graphical tool helps assess the performance of binary classification models. It visualizes the

trade-off between the model's ability to correctly identify true positives (TPR) and incorrectly classifying negatives (FPR) at various classification thresholds.

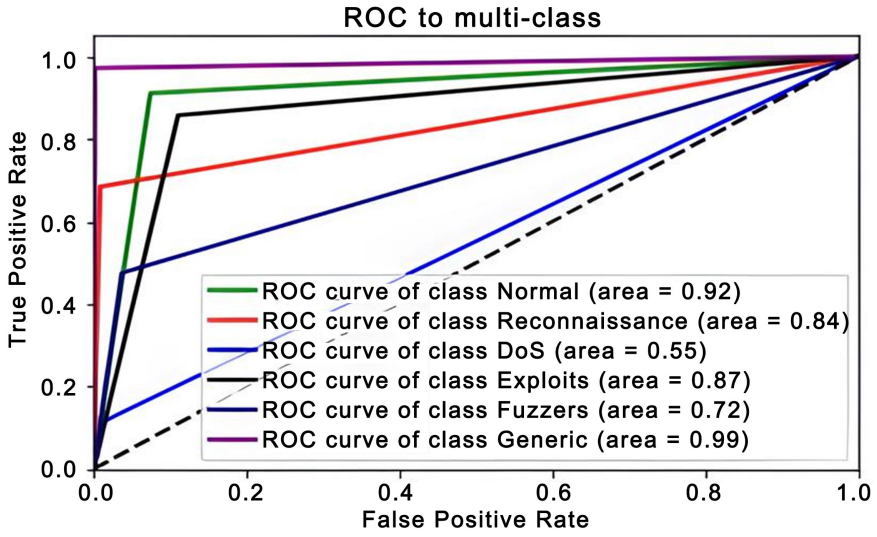


Figure 8. Roc Curves of class for DNN-XGBoost.

Case 2: LSTM

Table 4. Performance of LSTM-based attack types.

Attack Type	*LSTM-RF*	*LSTM-KNN*	*LSTM-XGBoost*
Normal	*92*	*91*	*92*
Reconnaissance	85	84	85
DoS	58	65	56
Exploits	85	80	87
Fuzzers	75	72	74
Generic	99	99	99

Table 4 shows the results of three classification models (LSTM-RF, LSTM-KNN, LSTM-XGBoost) for six types of attack (Normal, Reconnaissance, DoS, Exploits, Fuzzers, Generic). Each cell in the table represents the accuracy (in %) of the model for a given attack type. The LSTM-RF, LSTM-KNN and LSTM-XGBoost models show an average accuracy of 84.5%, 84.5% and 85.5% respectively. The LSTM-RF and LSTM-XGBoost models show lower variability in their results, with accuracies between 58% and 99%, while LSTM-KNN has higher but more variable accuracies, between 72% and 99%. The models perform differently for each type of attack:

- Normal: All models have high accuracy ranging from 91% to 92%.
- Reconnaissance: The models have a similar accuracy of 84% to 85%.
- DoS: LSTM-KNN has the best performance at 65%, followed by LSTM-RF at 58% and LSTM-XGBoost at 56%.

- Exploits: LSTM-XGBoost has the best performance at 87%, followed by LSTM-RF at 85% and LSTM-KNN at 80%.
- Fuzzers: LSTM-RF has the best performance at 75%, followed by LSTM-XGBoost at 74% and LSTM-KNN at 72%.
- Generic: All models are 99% accurate.

Table 5 presents the results of three classification models (LSTM-RF, LSTM-KNN, LSTM-XGBoost) evaluated on several performance metrics: precision, recall, F1-score, MCC (Matthews Correlation Coefficient) and MSE (Mean Squared Error). Each row of the table represents the values of these metrics for a given model.

Table 5. Performance of LSTM-based classification models.

Models	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	MCC (%)	MSE
LSTM-RF	*81.28*	*80.44*	*81.28*	*80.46*	*75.31*	*1.21*
LSTM-KNN	78.86	78.56	78.86	78.64	72.08	1.42
LSTM-XGBoost	81.21	80.14	81.21	79.90	75.27	1.27

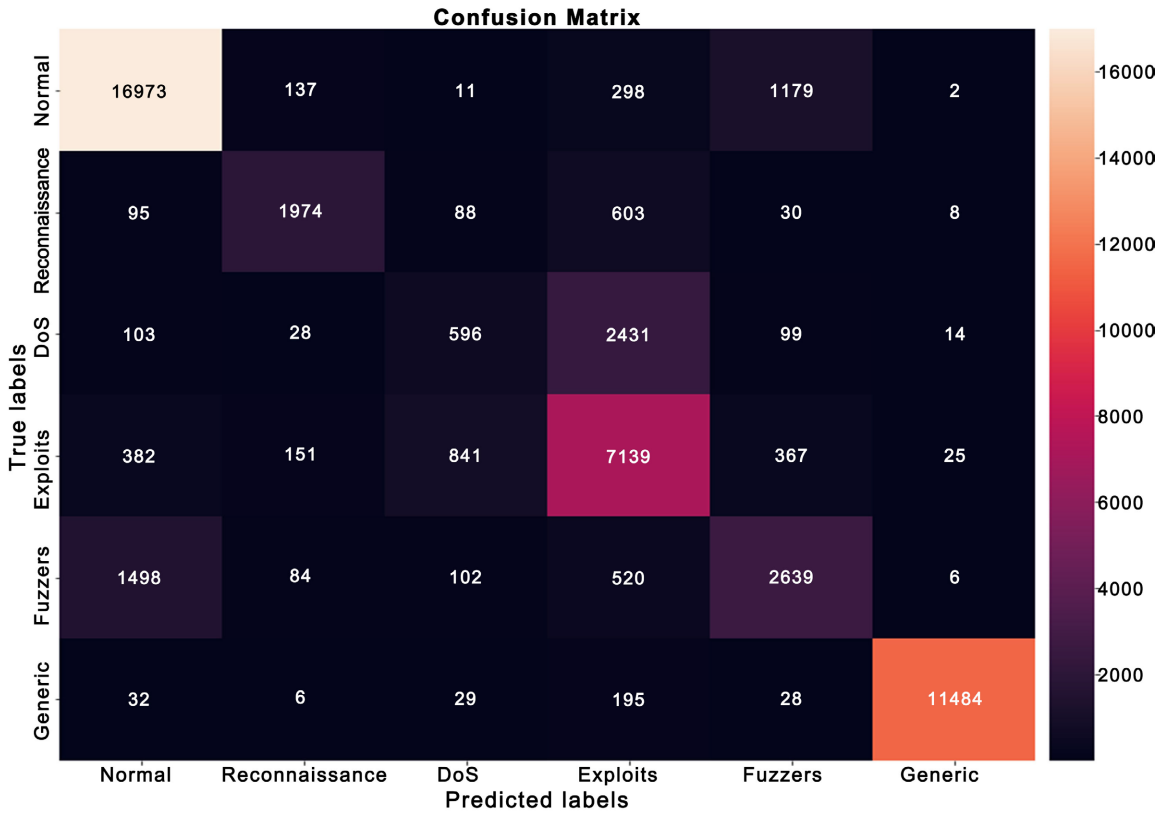


Figure 9. Confusion Matrix for LSTM-RF.

The LSTM-RF model appears to have the best overall performance, with the best values for precision, recall, F1-score, MCC and MSE. LSTM-XGBoost has slightly lower performance, while LSTM-KNN has the lowest performance of the

three models. However, the differences in performance between the models are relatively small.

Figure 9 show confusion matrix classification models LSTM-RF. The confusion matrix illustrated in the image is a tool used to evaluate the performance of a supervised classification model. It shows the number of correct and incorrect predictions made by the model for each class.

Figure 10 shows the ROC curve for evaluating the performance of a binary classification model for the LSTM-RF model.

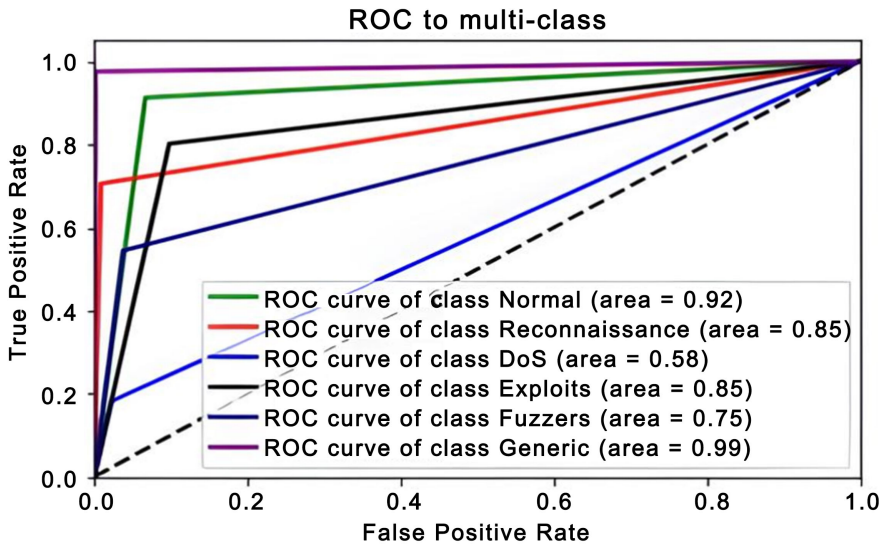


Figure 10. ROC Curves of class for LSTM-RF.

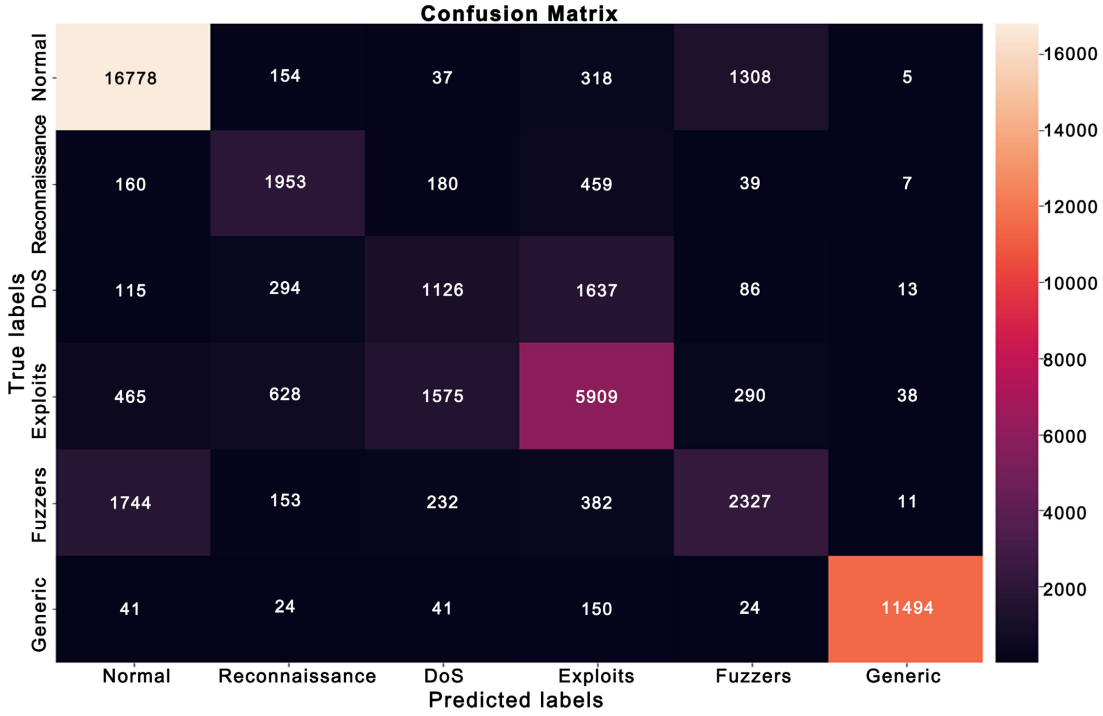


Figure 11. Confusion Matrix for LSTM-KNN.

Figure 11 shows the confusion matrix for the LSTM-KNN classification model. This table is a valuable tool for evaluating the performance of supervised classification models. It details the distribution of the model's predictions, classifying them as correct or incorrect for each class.

Figure 12 shows the ROC curve used to evaluate the performance of a binary classification model for the LSTM-KNN model.

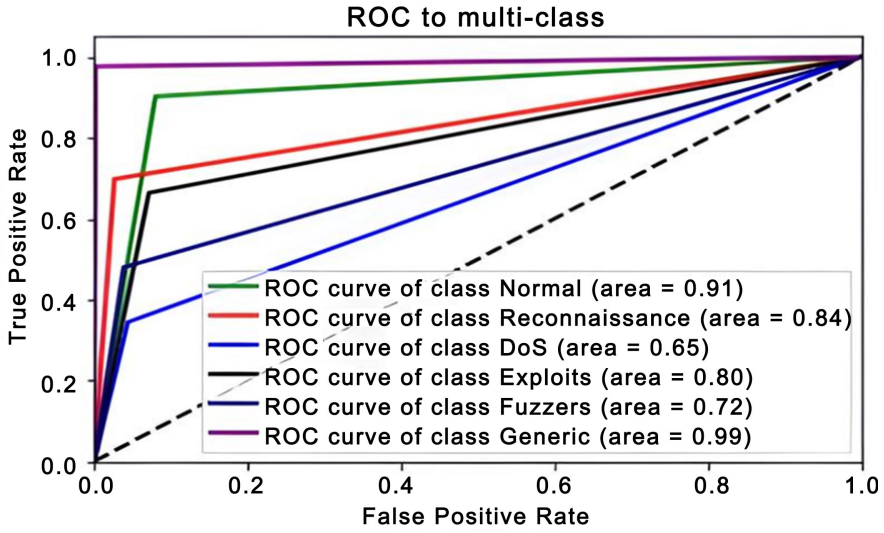


Figure 12. ROC Curves of class for LSTM-KNN.

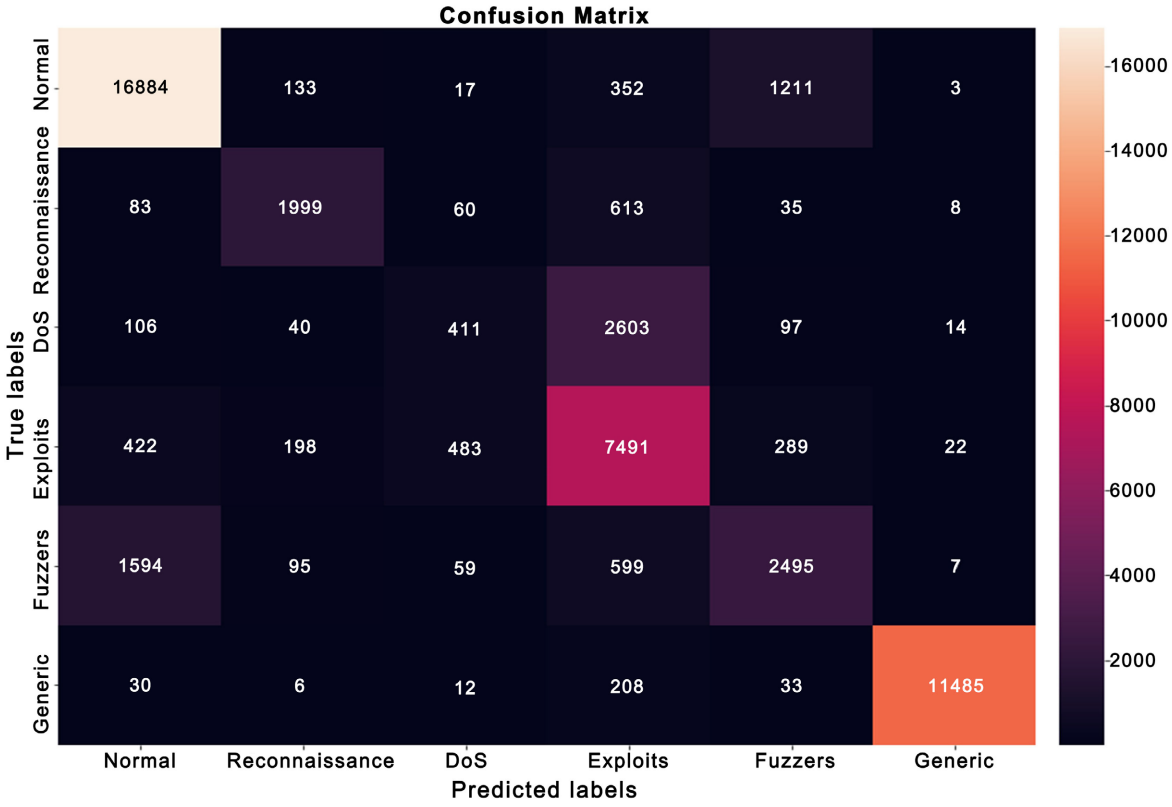


Figure 13. Confusion Matrix for LSTM-XGBoost.

Figure 13 shows the confusion matrix for the LSTM-XGBoost classification model

Figure 14 shows the ROC curve for the LSTM-XGBoost model. This graphical tool is used to evaluate the performance of binary classification models. It shows the trade-off between two key measures: the ability of the model to correctly identify true positive instances and the rate of misclassification of negative instances at different breakpoints in the classification.

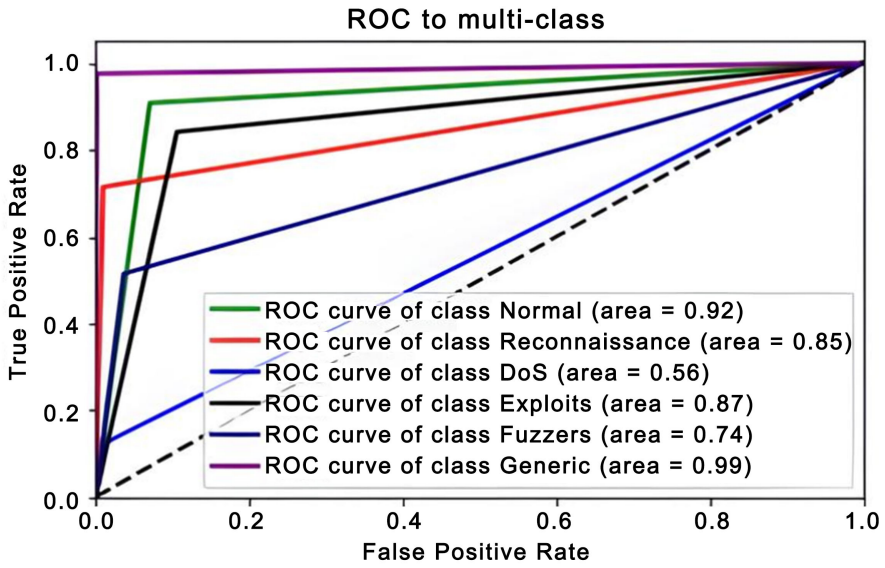


Figure 14. ROC Curves of class for LSTM-XGBoost.

5. Discussion

The results show that LSTM-based models (LSTM-RF, LSTM-KNN, LSTM-XGBoost) perform slightly better than DNN-based models (DNN-RF, DNN-KNN, DNN-XGBoost) in terms of overall accuracy, F1 score and MCC, indicating that LSTM-based recurrent neural networks are better suited to capturing temporal dependencies in attack detection data, thereby improving classification performance.

The DNN-RF model offers the best performance for the classification task under consideration. The combination of deep neural networks and random forests makes it possible to capture complex relationships within the data, significantly improving the model's ability to discriminate between different classes. The DNN-XGBoost model came second, also demonstrating excellent generalisation capabilities. The XGBoost algorithm, renowned for its robustness to overlearning and unbalanced data, contributes to the model's performance.

The results show that all the models have high performance, above 90%, for "normal" and "generic" attacks, indicating that they are well suited to these types of attack. However, performance varies more for other types of attack such as "DoS", "Exploits" and "Fuzzers", and we can see that some models are better suited than others to these more complex types of attack, as seen with LSTM-

XGBoost which performs better for "Exploits" attacks, while LSTM-KNN performs better for "DoS" attacks.

Although LSTM models generally offer better performance, DNN models have the advantage of lower computational complexity, with smaller model sizes, and are an important factor in applications where computational resources are limited, such as on embedded devices or mobile systems. The choice of model will therefore depend on the balance to be struck between detection performance and deployment constraints.

The LSTM models showed a slight advantage in terms of overall accuracy, particularly for more complex attacks such as denial of service (DoS) attacks and exploits. This superiority is explained by the ability of LSTMs to model the temporal dependencies present in data, which is particularly useful for detecting attacks that evolve over time.

Our models are capable of analyzing network traffic in real time to identify suspicious or unusual behavior. This makes it possible to detect attacks such as code injections, denial of service (DoS) or data exfiltration, aimed at compromising the integrity or availability of systems.

In comparison with the work of Ahmim *et al.* [30], we have developed new hybrid deep learning models (DNN-RF and LSTM-RF) specifically designed to detect DDoS attacks in IoT environments. Our experimental results show that our models significantly outperform those of Ahmim *et al.*, with overall accuracy rates reaching 81.32% and 81.28% respectively, compared with 80.75% for their model.

In terms of computing resources, all the models studied require a suitable hardware configuration. The size of the datasets, the complexity of the architectures and the hyperparameters chosen have a direct influence on memory and computing power requirements. The use of GPUs is generally recommended to accelerate training, particularly for the most complex models.

The choice of hyperparameters is crucial for optimising model performance. Parameters such as the learning rate, the size of the mini-lots and the depth of the trees for random forests have a significant impact on the speed of convergence and the quality of the results.

Our results highlight the complementarity of different neural network architectures and ensemble algorithms for the attack detection task. The choice of the optimal model will depend on the trade-off between performance, complexity and deployment constraints. The LSTM models seem particularly well suited to attacks that evolve, while the DNN-RF and DNN-XGBoost models offer excellent overall performance.

6. Conclusions

Our study shows that deep learning models, particularly those based on LSTM, are powerful tools for detecting attacks in IoT networks. These models offer high accuracy and robustness, particularly in the face of sophisticated attacks.

The choice of the optimum model depends on the specific requirements of the application and the trade-off between precision and complexity. For maximum accuracy and robustness, the LSTM-RF and LSTM-XGBoost models are the right choice. On the other hand, if better generalization in the face of different types of attack is paramount, the DNN-RF and DNN-XGBoost models may be preferred, even if their accuracy decreases slightly.

The prospects for this research are vast and exciting. The exploration of new neural network architectures, such as temporal convolutional neural networks (TCNs) or Transformer networks, could further improve the accuracy and robustness of attack detection. Similarly, the development of interfaces and tools to easily integrate deep learning models into existing cybersecurity architectures would facilitate their widespread adoption.

In the face of rapidly evolving technologies and cybersecurity threats, constant adaptation and innovation are required to maintain effective protection of IoT networks. Deep learning models, in particular those based on LSTM, offer a solid foundation for meeting this challenge and ensuring the security of critical IoT infrastructures in the years to come.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Nguyen, D.C., Ding, M., Pathirana, P.N., Seneviratne, A., Li, J., Niyato, D., *et al.* (2022) 6G Internet of Things: A Comprehensive Survey. *IEEE Internet of Things Journal*, **9**, 359-383. https://doi.org/10.1109/jiot.2021.3103320

[2] Kayode Saheed, Y., Idris Abiodun, A., Misra, S., Kristiansen Holone, M. and Colomo-Palacios, R. (2022) A Machine Learning-Based Intrusion Detection for Detecting Internet of Things Network Attacks. *Alexandria Engineering Journal*, **61**, 9395-9409. https://doi.org/10.1016/j.aej.2022.02.063

[3] Salman, O., Elhajj, I.H., Chehab, A. and Kayssi, A. (2019) A Machine Learning Based Framework for IoT Device Identification and Abnormal Traffic Detection. *Transactions on Emerging Telecommunications Technologies*, **33**, e3743. https://doi.org/10.1002/ett.3743

[4] Saba, T., Rehman, A., Sadad, T., Kolivand, H. and Bahaj, S.A. (2022) Anomaly-Based Intrusion Detection System for IoT Networks through Deep Learning Model. *Computers and Electrical Engineering*, **99**, Article 107810. https://doi.org/10.1016/j.compeleceng.2022.107810

[5] Madhu, B., Venu Gopala Chari, M., Vankdothu, R., Silivery, A.K. and Aerranagula, V. (2023) Intrusion Detection Models for IOT Networks via Deep Learning Approaches. *Measurement: Sensors*, **25**, Article 100641. https://doi.org/10.1016/j.measen.2022.100641

[6] Malathi, C. and Padmaja, I.N. (2023) Identification of Cyber Attacks Using Machine Learning in Smart IoT Networks. *Materials Today: Proceedings*, **80**, 2518-2523. https://doi.org/10.1016/j.matpr.2021.06.400

[7] Altulaihan, E., Almaiah, M.A. and Aljughaiman, A. (2024) Anomaly Detection IDS

for Detecting Dos Attacks in IoT Networks Based on Machine Learning Algorithms. *Sensors*, **24**, Article 713. https://doi.org/10.3390/s24020713

[8] Arjoune, Y., Salahdine, F., Islam, M.S., Ghribi, E. and Kaabouch, N. (2020) A Novel Jamming Attacks Detection Approach Based on Machine Learning for Wireless Communication. 2020 *International Conference on Information Networking* (*ICOIN*), Barcelona, 7-10 January 2020, 459-464. https://doi.org/10.1109/icoin48656.2020.9016462

[9] Hachimi, M., Kaddoum, G., Gagnon, G. and Illy, P. (2020) Multi-Stage Jamming Attacks Detection Using Deep Learning Combined with Kernelized Support Vector Machine in 5G Cloud Radio Access Networks. 2020 *International Symposium on Networks, Computers and Communications* (*ISNCC*), Montreal, 20-22 October 2020, 1-5. https://doi.org/10.1109/isncc49221.2020.9297290

[10] Arcangeloni, L., Testi, E. and Giorgetti, A. (2023) Detection of Jamming Attacks via Source Separation and Causal Inference. *IEEE Transactions on Communications*, **71**, 4793-4806. https://doi.org/10.1109/tcomm.2023.3281467

[11] Vatambeti, R., Damera, V.K., Karthikeyan, H., Manohar, M., Sharon Roji Priya, C. and Mekala, M.S. (2023) Classification of HHO-Based Machine Learning Techniques for Clone Attack Detection in WSN. *International Journal of Computer Network and Information Security*, **15**, 1-15. https://doi.org/10.5815/ijcnis.2023.06.01

[12] Kayode Saheed, Y., Idris Abiodun, A., Misra, S., Kristiansen Holone, M. and Colomo-Palacios, R. (2022) A Machine Learning-Based Intrusion Detection for Detecting Internet of Things Network Attacks. *Alexandria Engineering Journal*, **61**, 9395-9409. https://doi.org/10.1016/j.aej.2022.02.063

[13] Churcher, A., Ullah, R., Ahmad, J., ur Rehman, S., Masood, F., Gogate, M., *et al.* (2021) An Experimental Analysis of Attack Classification Using Machine Learning in IoT Networks. *Sensors*, **21**, Article 446. https://doi.org/10.3390/s21020446

[14] Alissa, K., Alyas, T., Zafar, K., Abbas, Q., Tabassum, N. and Sakib, S. (2022) Botnet Attack Detection in IoT Using Machine Learning. *Computational Intelligence and Neuroscience*, **2022**, Article 4515642. https://doi.org/10.1155/2022/4515642

[15] Xu, Q., Jia, X., Jia, B. and Liang, Y. (2022) IoT-Oriented Distributed Intrusion Detection Methods Using Intelligent Classification Algorithms in Spark. *Security and Communication Networks*, **2022**, Article 2842624. https://doi.org/10.1155/2022/2842624

[16] Pirayesh, H. and Zeng, H. (2022) Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, **24**, 767-809. https://doi.org/10.1109/comst.2022.3159185

[17] Wan, Z., Chang, Z., Xu, Y. and Šavija, B. (2023) Optimization of Vascular Structure of Self-Healing Concrete Using Deep Neural Network (DNN). *Construction and Building Materials*, **364**, Article 129955. https://doi.org/10.1016/j.conbuildmat.2022.129955

[18] Liang, H., Sang, Q., Hu, C., Cheng, D., Zhou, X., Wang, D., *et al.* (2023) DNN Surgery: Accelerating DNN Inference on the Edge through Layer Partitioning. *IEEE Transactions on Cloud Computing*, **11**, 3111-3125. https://doi.org/10.1109/tcc.2023.3258982

[19] Toumi, A., Cexus, J.C., Abid, M. and Khenchaf, A. (2023) Un réseauhybride CNN-LSTM pour la classification de navires à partird'une base frugale des images SAR. GRETSI-Groupe de Recherche enTraitement du Signal et des Images, Grenoble.

[20] Sun, Y., Dong, Y. and Chen, Y. (2024) Global Evapotranspiration Simulation Research Using a Coupled Deep Learning Algorithm with Physical Mechanisms. *Irrigation and Drainage*. https://doi.org/10.1002/ird.2942

[21] Bessaa, T., Bellal, F., Azzou, S.A.K., Boukredera, D. and Tafoukt, R. (2023) Analyse de l'Influence des Caractéristiques sur la Performance du ModèlePrédictif de la RétinopathieDiabétique. *Colloque sur les Objets et SystèmesConnectés* 2023, Mahdia, 22-24 June 2023, 1-5.

[22] Yang, J. and Guan, J. (2022) A Heart Disease Prediction Model Based on Feature Optimization and Smote-Xgboost Algorithm. *Information*, **13**, Article 475. https://doi.org/10.3390/info13100475

[23] Chen, T. and Guestrin, C. (2016) XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, 13-17 August 2016, 785-794. https://doi.org/10.1145/2939672.2939785

[24] Hashmi, A., Barukab, O.M. and Hamza Osman, A. (2024) A Hybrid Feature Weighted Attention Based Deep Learning Approach for an Intrusion Detection System Using the Random Forest Algorithm. *PLOS ONE*, **19**, e0302294. https://doi.org/10.1371/journal.pone.0302294

[25] Costa, V.G. and Pedreira, C.E. (2022) Recent Advances in Decision Trees: An Updated Survey. *Artificial Intelligence Review*, **56**, 4765-4800. https://doi.org/10.1007/s10462-022-10275-5

[26] Manzali, Y. and Elfar, M. (2023) Random Forest Pruning Techniques: A Recent Review. *Operations Research Forum*, **4**, Article No. 43. https://doi.org/10.1007/s43069-023-00223-6

[27] Ghunimat, D., Alzoubi, A.E., Alzboon, A. and Hanandeh, S. (2022) Prediction of Concrete Compressive Strength with GGBFS and Fly Ash Using Multilayer Perceptron Algorithm, Random Forest Regression and K-Nearest Neighbor Regression. *Asian Journal of Civil Engineering*, **24**, 169-177. https://doi.org/10.1007/s42107-022-00495-z

[28] Dalhat Mu'azu, N. and Olusanya Olatunji, S. (2023) K-Nearest Neighbor Based Computational Intelligence and RSM Predictive Models for Extraction of Cadmium from Contaminated Soil. *Ain Shams Engineering Journal*, **14**, Article 101944. https://doi.org/10.1016/j.asej.2022.101944

[29] Karamti, H., Alharthi, R., Anizi, A.A., Alhebshi, R.M., Eshmawi, A.A., Alsubai, S., *et al.* (2023) Improving Prediction of Cervical Cancer Using KNN Imputed SMOTE Features and Multi-Model Ensemble Learning Approach. *Cancers*, **15**, Article 4412. https://doi.org/10.3390/cancers15174412

[30] Ahmim, A., Maazouzi, F., Ahmim, M., Namane, S. and Dhaou, I.B. (2023) Distributed Denial of Service Attack Detection for the Internet of Things Using Hybrid Deep Learning Model. *IEEE Access*, **11**, 119862-119875. https://doi.org/10.1109/access.2023.3327620