An Enhanced Hybrid Deep Learning Model to Enhance Network Intrusion Detection Capabilities for Cybersecurity

[1]Abhijit Das, [2]Shobha N, [3]Natesh M, [4]Gyanendra Tiwary and [5]Karthik V
[1,5]Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India.
[2]Department of Computer Science and Design, Dayananda Sagar College of Engineering, Bengaluru, India.
[3]Department of Computer Science and Engineering, Vidyavardhaka College of Engineering, Mysuru, India.
[4]School of Computer Science and Engineering, RV University, Bengaluru, India.
[1]abhijit.das@manipal.edu, [2]shobha-csd@dayanandasagar.edu, [3]natesh.m@vvce.ac.in, [4]gyanendrat@rvu.edu.in, [5]v.karthik@manipal.edu

Correspondence should be addressed to Karthik V : v.karthik@manipal.edu

Article Info
Journal of Machine and Computing (http://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202404045
Received 10 April 2023; Revised from 12 October 2023; Accepted 12 March 2024.
Available online 05 April 2024.

Abstract – Recently, we have noticed tremendous growth in the field of Information Technology. This increased growth has proliferated the use of new technologies and continued advancement of networking systems. These systems are widely adopted for real-time online and offline tasks. Due to this growth in information technology, maintaining security has gained huge attention as these systems are vulnerable to various attacks. In this context, an Intrusion Detection System (IDS) plays an important role in ensuring security by detecting and preventing suspicious activities within the network. However, as technology is overgrowing, malicious activities are also increasing. Moreover, legacy IDS methods cannot handle new threats, such as traditional signature-based methods requiring a predefined rule set to detect malicious activity. Also, several new methods have been proposed earlier to address security-related issues; however, the performance of these methods is limited due to poor attack detection accuracy and increased false positive rates. In this work, we propose and compare different deep-learning (DL) models that can be used to construct IDSs to provide network security. Details on convolutional neural networks (CNNs), Multilayer Perceptron (MLP), and long short-term memories (LSTMs) are introduced. A discussion of the outcomes achieved follows an assessment of the proposed DL model known as the FOA-CNN-LSTM technique. Comparisons are made between the suggested models and other machine-learning methods. This work presents a deep-learning approach based on hybrid CNN-LSTM with Fruit fly Optimization Algorithm (FOA) by ensemble techniques to distinguish between normal and abnormal behaviors.

Keywords – Intrusion Detection System, Cyberattacks, Fruit fly Optimization Algorithm, LSTM, CNN, DNN, RNN.

I. INTRODUCTION

Due to its ability to understand existing data, DL-based system is currently a popular topic in AI. As academics in other domains have discovered encouraging outcomes from deep learning, they have begun to focus on cybersecurity. When we looked back just a few years ago, the idea of deep learning was still around, but it wasn't widely deployed because of the enormous volume of data, computing power, and other resources it required. It is possible to divide the learning algorithms into supervised, unsupervised, and semi-supervised categories. All three types of learning can use deep learning, a specialized branch of machine learning based on a more comprehensive neural network.

On the other hand, there needed to be more computing power to make deep learning model training feasible. Furthermore, vanishing gradient issues plagued earlier deep-learning methods. The lengthy training period required by deep learning techniques is a drawback of the process. However, DL methods necessitate a large quantity of training data for optimal performance. The advent of considerably speedier technology and the introduction of GPUs as processing units has brought deep learning to practical applications. All the major IT firms currently use deep learning technology in their products and services.

Recently more focus has been placed on Intrusion Detection Systems (IDSs) to protect critical data within a network infrastructure. In most cases, hackers can find a way into a protected network by exploiting vulnerabilities or using intelligent

attacks. Predicting attacks from normal packets is difficult, time-consuming, and technically demanding in such a setting. Accordingly, numerous algorithms' learning and training capabilities have been investigated. Existing Intrusion Detection systems, however, needed help to learn the complex feature automatically to achieve the required performance levels. As a result, a novel Intrusion Detection strategy is proposed here that combines the Deep Learning approach with the FOA strategies. FOA is a population-based strategy used to improve the effectiveness of the neurons in the hidden layers of Deep Learning methods. In addition, this work presented a CNN, MLP, LSTM, CNN-LSTM, and FOA-CNN-LSTM model for detecting cyberattacks and compared their performance. All of the deep learning models use the most recent CICIDS2018 datasets. In this work, we also evaluated the DL model against some of the most popular machine learning algorithms utilized in cyber security.

II. WORK IN THIS AREA

The term "deep learning (DL)" refers to a specific type of machine learning in which a multi-layered neural network (NN) is used to solve problems. These NN "learn" from extensive datasets to mimic human brain activity, albeit they are still far from brain supremacy. A single-layer NN can still produce approximations, but more complex networks with hidden layers can improve accuracy. Many independent deep networks are used to construct a deep learning model. The high efficiency of deep learning on vast volumes of security data is its main benefit over shallow ML. Research into deep learning-based IDS has seen rapid growth from 2015 until the present. DL methods can automatically learn characteristic representations from the original data, eliminating the requirement for time-consuming feature engineering. As a result, DL methods require less information processing to achieve the same degree of performance. DL methods outperform more conventional ML models when confronted with big datasets. Some of the most widely used DL approaches in cyberattacks are autoencoders, LSTM, CNN, GAN, DBN, DNN, and RNN, among the most effective learning methods.

When the information is complicated and includes a significant level of dimensionality, deep learning (DL), a subfield of machine learning (ML), performs exceptionally well in generalizing to new examples [1]. Furthermore, deep learning allows for the efficient training of nonlinear systems on massive datasets. A model that can generalize to assaults that aren't reflected in the available labelled data is crucial in the field of NIDS, where massive amounts of data must be processed. The ideal system would be able to generalise and perform well in a novel, unseen network contexts. However, even if it couldn't, it might still be employed in a machine learning process as a transfer learning phase when fed data from another network.

Table 1: Survey of Attacks and Detection Approach Applied by Researchers

Year	Vulnerability type	Detection Method	Features	Dataset	Accuracy	Ref
2020	Web Traffic	Random forest Algorithm, Gradient boosting Machine	Generalizability	CSIC-2010v2 and CICIDS-2017, NSL-KDD,UNSW-NB15	94.46%	[4]
2020	Host or Network Attacks	Hybrid of Decision Tree And KNN	Network Security	NSLKDD, KDD-Cup99	99.33%	[5]
2019	Ping of death, denial of Service,Host Attack	Expectation Maximization (EM),S VM,Naïve bayes Bayes,KNN,HNB	Improves the stength of contemporary method	NSI-KDD,KDD-Cup99, ADFA-LA and AFDA- WD	78%	[6]
2019	U2R and R2L Attacks	FAIS,FCAAIS	Scope of detecting novel attacks	NSL-KDD	88%	[7]
2018	Denial of Service,Distributed Dos	ANN,K-Means,SVM	Categorizing network attacks based on source	KDD-99,NSL-KDD	97.25%	[8]
2020	Packet-Level Attack	LSTM,OHE	Packet-level feature extractio improves real-time speed	ISCX2012,USTC-TFC2016,CICIDS2017	99.74%	[9]
2019	User to root attack,Remote to local attack,DoS,Probe	LSTM with,LR,NB,KNN,SVM	Enhanced Performance	KDDCup99,NSL-KDD,UNSW NB- 15,WSN-DS,CICIDS2017,ADFA-LD,ADFA-WD	93.50%	[10]
2021	Network Traffic	Ensamble Approach with C4.5, Random Forest,(Forest PA)	High Accuracy and Efficiency	NSL-KDD,AWID,CIC-IDS2017	99.52%	[11]
2018	Denial of Service,Distributed Dos	NB,Decisiontable and Stochastic gradient descent	High Accuracy	NSL-KDD	99.93%	[12]
2018	network Traffic	SVM,Random Forest Algorithm	Performance Improvement	NSL-KDD	99.33%	[12]

Table 1 shows survey of Attacks and detection approach applied by researchers. Although deep learning has been more well-known in recent years, it has been present for quite some time, with its origins going back to the 1940s [2]. Motivated by the computational methods of the human brain, a few of the earliest deep learning techniques gave rise to the terms "artificial neural network" (ANN) and "computational node," respectively. Although the neuroscientific viewpoint is still regarded as a significant motivation for deep learning, it is no longer used as the major basis for the subject. This is because complete knowledge of the brain's internal workings and techniques remains elusive. In the realm of "computational neuroscience," this is a topic that is constantly being studied and expanded upon. While brain-inspired models like the perceptron and neuron have affected deep learning's structure and direction throughout the years, they are by no way a hard and fast rule. Instead, the multi-level composition is a foundation of current deep learning . The rebirth of deep learning in the 2000s can be partially attributed to the explosion in processing capacity and data availability. Deep learning algorithms perform exceptionally well when a vast amount of information is available. As of 2016 [3], a decent general principle for supervised deep learning algorithms is to have at least 5,000 labelled samples per class. They will also outperform humans when trained with a dataset containing at least 10 million labelled samples. Moreover, some human analysts are experts in their fields and can offer constant input to a training system. One can use deep learning algorithms that are well adapted to this massive data set to uncover malicious behaviour hidden among the noise of a corporate network's traffic and system logs.

III. PROPOSED METHODOLOGY

The suggested method uses a neural network to analyze a CSE-CIC-IDS2018 dataset made up of modern attack vectors. This work created a method for identifying anomalous network traffic using deep learning techniques based on a hybrid CNN and LSTM model with FOA [13][14]. Notably, the timestamp values of each network flow are included, making the dataset better realistic. Many studies have used machine learning to justify their findings, yet this data has been mostly ignored. However, in the case of cyberattacks, a correlation of traffic flows as per their timing is crucial for detecting irregular activity. Because of its effectiveness in solving sequence issues, pattern recognition, audio recognition, text generation, text analysis, visual description generation, and character recognition, the CNN-LSTM-based deep learning techniques was chosen for this work.

In this work, we employed a CNN-LSTM-based FOA optimization technique to learn attack patterns and evaluate its performance in detecting malicious behavior. The FOA algorithm found the best smell by optimizing the model's hidden neurons and hidden representation via deterministic mapping with new values. By contrasting the ensemble CNN-LSTM model's performance in this way, we were able to assess the significance of features retrieved from the network to identify the attack behavior.

Datasets

In light of the ever-increasing sophistication of cyberattacks, it is crucial to train a model using the most recent realistic data available to construct an effective IDS. CSE-CIC-IDS2018 is a publicly available dataset [15] and it was chosen for this study. During the tests, six distinct types of attacks like brute-force, bott, Infiltration, web and DDoS were implemented. This dataset's 83% percentage of benign traffic is one of its realistic specialties. It is essential to make use of the assumed normality of most of the traffic in a real-world configuration to train the IDS. It will improve the system's capability to differentiate benign and malicious traffic, reducing the number of FP and improving accuracy. Hence reducing the number of FP while improving accuracy. More or less, every intrusion originated out of a machine (or machines) outside the defended system. Five distinct subnets were set up to represent the corporation's various departments and server rooms, just as they would be in a real network. All fourteen attack types are represented in this dataset's network traffic distribution. This dataset also contains gathered raw network traffic statistics organized daily into 10 CSV files. Information like time, no. of pkts, no. of bytes, and pkt length are sent in the fwd. and reverse paths, which significantly aids in detecting cyberattacks.

Experimental setup

The Anaconda Individual Edition Python distribution platform and Python, TensorFlow, Keras, and Scikit-learn were used to experiment. The CNN-LSTM model was developed and trained with Tensorflow and Keras. The hardware used in the experiments included a CPU (IntelR CoreTM i7), RAM (32 GB), a graphics processing unit (NVIDIA GeForce GTX), and an OS (Windows 11). In order to train on a graphics processing unit, Nvidia Cuda and CuDNN were installed and set up.

Preprocessing

This work employs a multi-class classification strategy to anticipate safe traffic against 14 distinct attack types, with the goal of outperforming conventional intrusion detection systems**. Fig 1** depicts the data preprocessing method designed for the experiment. Data Initiation, Data Preparation, and Split and Normalize were the three primary Phases of Preprocessing. To do this, we employ the procedure depicted in **Fig 1**. There are a total of ten daily CSV files in the dataset, and they have all been consolidated into a single file. Most prior efforts in this area dealt with particular forms of assault that were documented in isolated CSV files. However, since CNN-LSTM is the chosen method for DL, historical input is also included during the training process. To accurately categorize normal flow and recognize the abnormal activity, it gathers

as many examples as possible, as they understand the natural behavior of the flow. In addition, attacks like DDoS, brute force, bot, and infiltration depend on prior traffic. As a result, we use the entire preprocessed dataset to train the model. The duplicate rows were eliminated when the dataset was merged into a single CSV file. Next, the data is reorganized based on the "Timestamp" field. As was previously mentioned, the LSTM model relies heavily on previous data. Each epoch's batches were fed into the model, and the process was repeated. Timely network traffic analysis allowed more accurate detection of the attacker's tactics. The columns "Src IP", "Src Port", and "DstIP" were removed so that predictions may be made without reference to the original socket. We are keeping "Dst Port" since it helps us spot certain kinds of assaults, such as brute-force FTP and SSH attempts, web attacks, application-layer DoS attempts, and more.

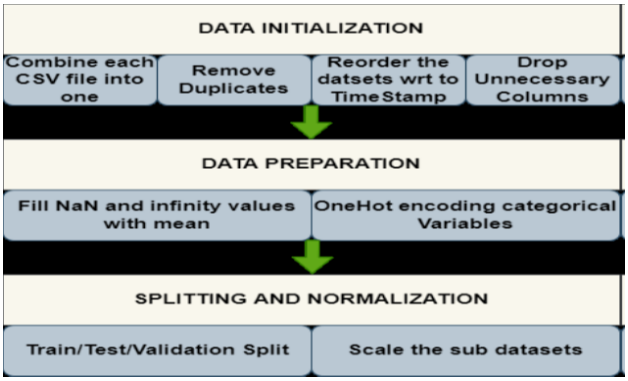

Fig 1. Data Pre-Processing

Further to the preceding, it was found that a few columns contributed little to the overall classification. The bandwidth_push_flag and bandwidth_urgent_flag columns are consistently found to have the same value (0). Furthermore, the flow_identifier produced a distinct value, and is not useful as a feature for attack detection because it is not shared throughout flows. Previously, the dataset was rearranged using the "Timestamp" feature. However, this column is also removed before training as the model does not need to distinguish between attack and benign traffic based on their timing. After eliminating the unnecessary attributes, the number of features has become 76 from 80. The splitting and Normalizing Phase follows the Data Initialization Phase to ready the data for further processing. The Flow bytes per second and Flow packets per second both have infinite values, which have been changed with NaN. The dataset was split into an independent variable array (labelled "y") and a dependent variable array (labelled "X") in NumPy. In this case, we pick as a dependent only the "Label" column describing the attacks. The resulting inverted 'X' coordinates are 15677150, 76, and those for 'y' are 15677150. Then the mean value from the corresponding column was substituted for the NaN value of X using the Simple Imputer function in sci-kit-learn. If we provide a Simple Imputer with a missing value strategy, it will use that to fill in the missing values in the dataset. Mean, median, most common, and constant are all possible choices. For this task, the model fell back on its default "mean" method. All independent variables were measured in numerical form. Categorical and string forms of the dependent variables were used. However, since the ML model requires numbers, it is important to transform all values into numbers before beginning the training phase. Accordingly, the Categorical Encoder converts the "Label" column (y) into a numerical array. All good traffic is represented by "0.0" on the y-axis, whereas all bad traffic is represented by "1.0" for binary classification purposes. OneHotEncoding is chosen as the method to use for multi-class categorization. Categorical characteristics are converted into a numerical array for use with OneHotEncoder.

In contrast to the other well-known method, LabelEncoder can store information as one hot numeric array of 0s and 1s. Machine learning models, including CNN and LSTM, generally emphasize more extreme values during training. Since all attack types are equally important for intrusion detection in multi-class classification, this circumstance takes on further significance. To split and normalize the data, we employ sklearn.model_selection.train_test_split. 20% to test and 80% to train have chosen. The result of the sub-dataset ratios is 64% to train, 20% to test, and 16% to validate. Tables 1 and 2 show the class distribution.

After splitting, sub-datasets are normalized between (0,1). It is accomplished by employing the "MinMaxScaler" function. It has been chosen since it does not alter the original data in any way, allowing us to keep the distribution's natural structure. The formula for minmax scaler has given in equation 1.

$$X_{minmax} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Model Building

This section discusses the model-building approach followed to design an FOA-CNN-LSTM along with the MLP, CNN, LSTM, and CNN-LSTM to identify malicious activity in a network. Some recent studies have investigated the viability of MLP, CNN, and LSTM in the intrusion detection domain. Some instances where MLP has proven to achieve remarkable outcomes are provided in [16][17]. The MLP's strengths include its ability to learn from imperfect data, fault tolerance,

simplicity (in construction and maintenance), computational speed, responsiveness to noise, and capability to learn and generalize. We also make use of the Convolutional Neural Network DL model. CNN's ability to automatically identify crucial elements, even without human intervention, is one of its main selling points. The use of CNN to identify DDoS attacks has been documented in papers such as [18] and [19]. The sparsity of connections describes how CNN's output values depend on a relatively small input value.

When training, the network's overall size can be kept relatively small thanks to the sparsity of its connections. RNNs are helpful since they are not constrained by the input length and can make better predictions by considering the temporal context. Long short-term memory (LSTM) aims to optimize neural network weights via vanishing gradient descent to prevent long-term reliance issues. Different values for the model's hyperparameters, like the learning rate, are tried out, and the optimal ones are discussed here. Considering that our data can be divided into two categories—"normal" and "attack"—the final layer of our models is highly dense and uses a sigmoid activation function. We have also developed LSTM, a DL approach for capturing long-term dependencies. A basic MLP, CNN, and LSTM model still produces excellent results. Still, it's worth noting that when LSTM models are paired with prior CNN models by using the FOA optimization algorithm, the result increases even further.

MLP Model

Multilayer Perceptron model used in this study is displayed in **Fig 2**. Transformation of MLP model's shape is not require before feeding it into the model as dataset used by MLP model is in the matrix format. The suggested model has an input layer, followed by three dense layers. Each layer's output feeds into the next layer's input. In order to prevent the system from overheating, a single dropout layer is added. After the dropout layer's output is processed, it is sent on to the fully connected layer, which in turn feeds the dense layer's sigmoid function.

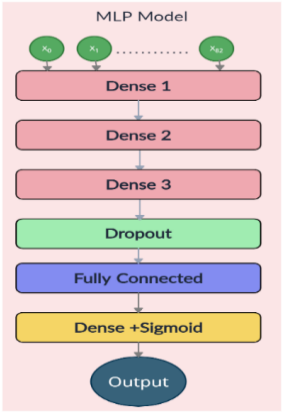

Fig 2. Multilayer Perceptron model

CNN Model

The design of the CNN model used is depicted in **Fig 3**. Convolutional, pooling, and dense layers are the three types of primary layers found in any CCN model. Input data is typically in a 3D shape (batch, steps, channels), and 1d-CNN will convert the dataset to a 3D shape as needed. The dataset includes 83 characteristics (with the last label), and we used the transform function with the parameters (data.shape[0], data.shape[1], 1) as well as transmitted the data into the network with an input shape of 82,1. We then activated the network with relu. An additional max pooling layer will only retain the highest-scoring features and reject any others. There is a dense sigmoid activation function in the last layer.

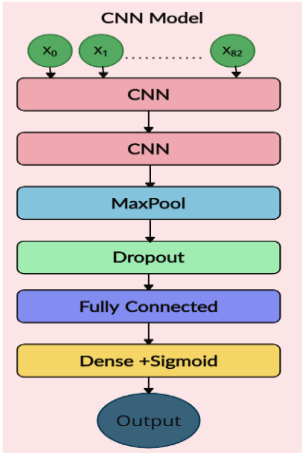

Fig 3. CNN Model

LSTM Model

LSTMs are a variant of RNNs in which nodes in the same layer are interconnected to eliminate irrelevant information and facilitate long-term memory storage. **Fig 4** depicts the LSTM model's flowchart, which allows for a 3D dataset to be input in the form of a batch, steps, and channels. First, an LSTM layer with 128 adam-activated kernels and then a dropout layer with a dropout rate of 0.5 make up this model. A dense layer with a sigmoid function receives input from the dropout layer to distinguish between normal and malicious behavior.

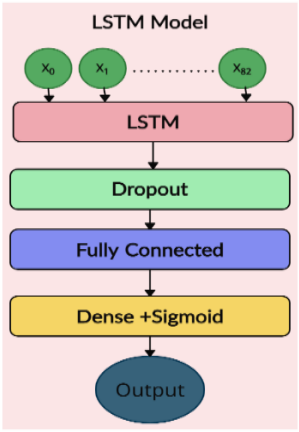

Fig 4. LSTM Model

CNN-LSTM Model

This model begins with a CNN layer using the relu activation function and continues with an LSTM layer using the adam activation function. The proposed hybrid CNN-LSTM model has shown in **Fig 5**. The dataset is transformed into a three-dimensional form (batch, steps, channels) for use with both CNN and LSTM. Because CNN produces output in a 3D space, the data does not need to be reshaped before being fed into the LSTM model. Parameters are similar to convolutional neural networks and Long Short Term Memory.

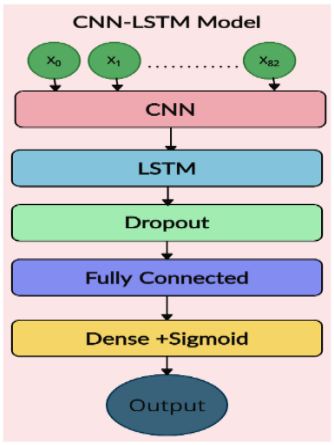

Fig 5. CNN-LSTM Model

FOA-CNN-LSTM Model

The details of the suggested architecture for attack detection with the CNN-LSTM model are shown in **Fig 6**. The primary goal of this study was to tune the most sensitive hyperparameters of a CNN-LSTM network to automatically generate a highly accurate CNN-LSTM model throughout the model-building process. Therefore, we carefully regulate these facets of the model-building process, which significantly aids in improving the time series forecasting model's predictive capability. In order to determine if a behavior is benign or malicious, this research proposes an ensemble mechanism that combines the CNN-LSTM with FOA.

The proposed methodology consists of two distinct but interconnected steps: data preprocessing and hyper-parameter tuning with an optimized CNN-LSTM model. The first processing module, data preparation, includes the submodules of data initialization, data preparation, splitting, and normalization, all of which were covered in the previous part. After that, we move on to fine-tuning the CNN-LSTM model's hyper-parameters in the second processing module. In this part, we will adjust the CNN-LSTM hyper parameter like batch size, the number of hidden neurons, the number of hidden layers, and the number of epochs. This network is then validated by finding the best smell with optimized hidden neurons, optimized weight, and hidden representation through deterministic mapping.

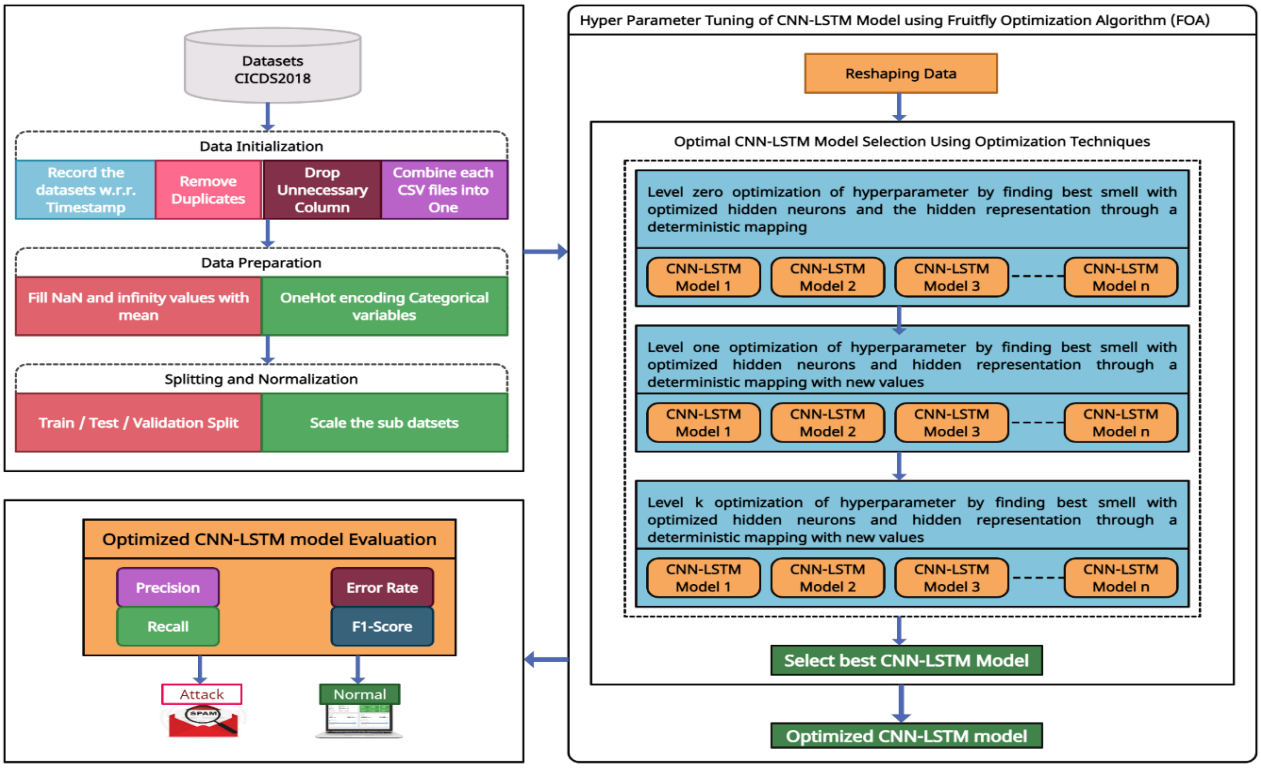

Fig 6. The proposed CNN-LSTM hybrid model with FOA Optimization Techniques

The proposed CNN-LSTM hybrid model with FOA optimization techniques combines the strengths of both networks by leveraging CNN for feature extraction and LSTM for prediction while taking advantage of FOA to improve model performance further. The core component of the proposed method is a CNN-LSTM architecture. The convolutional layer, pooling layer, and flattening are all parts of the CNN module. The m-layer configuration for the Conv2d convolutional layer has been selected. CNN's feature extraction can be tailored to data from varying time periods by adjusting the size of the convolution kernel. The convolution kernel is scaled to n X n to make the most of the information ready. The pooling layer size is n X n, and the maximum number of layers is m. The research improves training effectiveness by inserting a batch normalization layer before the pooling layer. After that, flatten is used to reduce the file size so that global features can be extracted. The information is then fed into the LSTM module for forecasting purposes. Experiments have shown that boosting the model's depth by adding more LSTM network nodes enhances the model's ability to predict. The number of neurons in each layer of the LSTM is set by the proposed FOA optimization algorithm 1. Dropout is utilized in LSTM to mitigate the possibility of overfitting.

Moreover, finally, Dense will spit out the attack prediction value as a vector in the given format. Compared to the CNN-LSTM method, the FOA-CNN+LSTM algorithm suggested here performs better. With regards to the CNN-LSTM method, its usefulness, particularly in the realm of prediction, has been established by numerous articles. Because of its special benefits for time series prediction, LSTM is used as input in the FOA-CNN+LSTM method. LSTM will prevent gradient fading while predicting time series over lengthy periods of time. However, most studies rely on manual selection methods like grid search to choose the best parameters for their CNN+LSTM models, which is both time-consuming and inefficient.

Consequently, this study opts for the FOA approach for parameter selection because it is more time-efficient in terms of finding the optimal smell with optimized hidden neurons, weight, and hidden representation via a deterministic mapping with new values. Here, we use FOA to determine the optimal combination of convolution kernel size, number of neurons per LSTM layer, and weight for optimization. After training the model a certain number of times or until some other criterion is met, the best value can be used to forecast an attack.

Pan presented the Fruit fly optimization algorithm (FOA) in 2012 [13] as a population-based approach to optimization. The algorithm's development was inspired by the way FOA forage for food. In particular, it can detect the smell of the food source from great distances. As a result, the FOA has lately been used to fine-tune the settings of a wide range of real-time programs[18][19]. As part of this effort, we use Fruit fly to fine-tune the neurons in CNN-hidden LSTM layers. The CNN-LSTM is taught to use several hidden layers to get the most useful information from raw input. During this training phase, the FOA algorithm is used to fine-tune the CNN-LSTM's hidden neurons. Even further, the backpropagation neural network is used to categorize the assaults. The number of iterations is 200, and the learning probabilities are 0.01 and 0.001. Weights and thresholds of the CNN-LSTM model are initialized at the population location (x0, y0) of fruit flies. FOA runs for 1000 iterations, with a population size of 20. Individual searches can be set to travel in a predetermined direction over a

predetermined distance randomly. We use a fruit fly optimization approach to determine the best values for the weights and the cutoffs. The CNN-LSTM model is then updated with the optimal value in order to perform repeated calculations.

In order to ensure an objective evaluation of results, testing input is not utilized while training. When fed training sets, the FOA-CNN-LSTM generates an error, which is then used by the optimizer, which employs a backpropagation technique, to fine-tune the network's parameters. With more training cycles, GA-CNN-LSTM will produce more precise predictions. After the FOA-CNN-LSTM has been trained, it is fed testing data, and its performance is judged based on how closely the test results match the true outcomes. Algorithm 1 describes the proposed FOA-CNN-LSTM technique.

Overfitting can occur when there needs to be more training data or when there is too much training. Overfitting can be avoided by a variety of techniques, including regularization, data augmentation, Dropout, dropconnect, and early termination. In this study, we employ a dropout methodology.

Algorithm 1. IDS using hybrid CNN-LSTM with FOA Optimization

Step 1: Start by loading the data, which has been separated into classes for training and testing.

Step 2: Set the starting conditions for the fly optimization process, including population size, number of iterations, fruit fly position, random search directions, and starting search length.

Step 3: Train CNN-LSTM with random hidden values and compute the accuracy

Step 4: Compute the cityblock distance of the Fruitfly from the origin

$$D_i = \sum_{i=1}^{n} |x_i + y_i| \tag{2}$$

Step 5: Find the smell concentration judgement value

$$s_i = \frac{1}{D_i} \tag{3}$$

Step 6: Compute the smell concentration of the individual Fruitfly

$$Smell_i = Fitness(S_i) \tag{4}$$

Step 7: Identify the position of the best smell concentration

$$[bestSmel_{Index}] = max(Smell_i) \tag{5}$$

Step 8: Go to step 3 until convergence

Step 9: Train CNN-LSTM with optimized hidden neurons and find the hidden representation through a deterministic mapping

$$Encode_1 = \sigma_r + (W * I + b) \tag{6}$$

Step 10: The latent representation from step 8 is passed as an input to the successive CNN-LSTM

$$Encode_2 = \sigma_r + (W * Encode_1 + b) \tag{7}$$

Step 11: Train Back Propagation Neural Network with the encoded features to classify the attacks

Step 12: The trained CNN-LSTM is updated with the optimal weights and thresholds.

Step 13: Evaluate the performance of the hybrid deep CNN-LSTM performance using the test sets and compute the error.

Step 14: From the ensemble of model select the best CNN-LSTM model

We have also tested and compared several well-known machine learning algorithms alongside the aforementioned deep learning models. SVM is widely utilised in supervised classification and regression problems. In order to classify a data point correctly in n dimensions, SVM looks for a decision limit. Support vector machines (SVMs) pick out the hyperplane's extrema or support vectors. We have tested the Naive Bayes classifier based on the Bayes theorem and employed it in supervised classification. Bayes classifier's speed and accuracy as a machine learning method for prediction is a significant plus. It relies on the object likelihood to forecast the final result. Ensemble learning, upon which Random Forest is founded, combines multiple classifiers to find a solution to a complex problem. Subsets of a dataset are used to increase the predicted accuracy of the dataset by feeding it to several decision trees. Each decision tree's prediction is added together, and the final result is forecast using the predictions with the most votes.

IV. EXPERIMENTAL ANALYSIS &RESULTS

Our proposed method is demonstrated here with experimental results employing FOA-based optimization tools and their performance measured. Additionally, the improved outcomes acquired through CNN-LSTM network optimization have been assessed. Therefore, employing FOA approaches, it has been demonstrated how to construct an optimised CNN-LSTM network by gradually increasing the hyper-parametric values depending on the lowest prediction error at every iterative level, starting with level-0 optimization and culminating through level-k optimization. This analysis is concluded using CICDS 2018 datasets.

Table 2 displays the results of a comparison between CNN-LSTM network hyper-parameter tuning at level 0 and level k. Experiments were run to find the best possible settings for the n-LSTM model's hyper-parameters and network topology, and the results are shown below. The results demonstrate the efficiency of an assembled optimization method for exploring a CNN-LSTM network model.

Table 2. Optimized hyperparameter Using FOA from level 0 to level k.

Sl.	There are n LSTM models in the initial population, and they are all optimized at hyperparameter level 0.					There are n LSTM models in the initial population, and they are all optimized at hyperparameter level k.				
1	**Hidden Layer**	**Hidden Neurons**	**Batch size**	**Epochs**	**Error rate**	**Hidden Layer**	**Hidden Neurons**	**Batch size**	**Epochs**	**Error rate**
2	2	45	5	304	0.28	2	41	8	277	.273
3	8	70	21	244	0.40	5	53	17	247	.394
4	4	22	16	64	.39	3	31	14	157	.381
5	7	65	20	440	.42	**2**	**20**	**10**	**182**	**.141**
6	1	99	15	186	.37	.594	1	70	218	.366
7	4	85	4	212	.30	3	63	8	231	.285
8	2	40	12	250	.22	2	35	13	256	.218
9	7	97	7	266	.45	5	69	9	258	.441
10	1	77	8	441	.36	1	59	10	346	.345
…	…..	……	……	…….	…….	…….	……..	…….	……	……
n	…..	……	……	…….	…….	…….	……..	…….	……	……

To begin, binary classification is tried out for this dataset, just like many other works have been done previously utilizing machine learning algorithms on the same dataset. In order to make predictions between two possible outcomes, binary classification is employed. The goal here is to determine whether the flow is malicious or benign. Model development and preprocessing follow the same procedures as described in the introduction. After determining that 10 epochs provide the best trade-off between training loss and accuracy, this binary training method employs this parameter setting. The accuracy and loss of training over time are displayed in **Fig 7** and **Fig 8**. In **Table 3**, we can see the outcomes of our performance analysis.

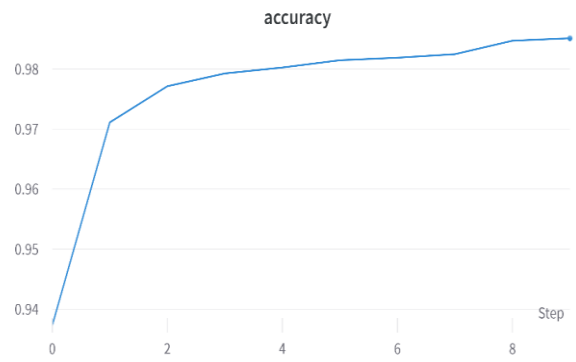

Fig. 7 Training Accuracy (Accu) for a Two-Class Problem

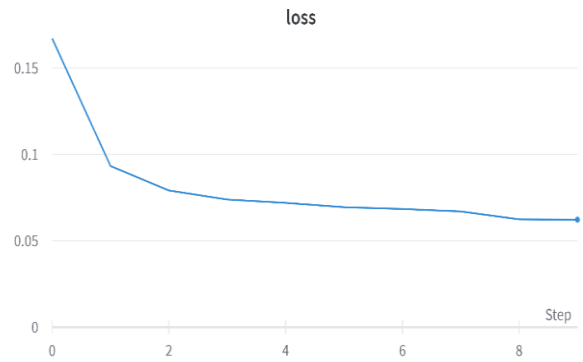

Fig. 8 Training Loss for a Two-Class Problem

Table 3. Performance Evaluation on Two-Class Classification

Metric	**Outcomes**
Accu	0.9932
Recall	09933
Precision	0.985
F1-Score	0.985
G Mean	0.961
Balanced Accu	0.949

Fig 9 shows the confusion matrix. The TN ratio is highest compared to the other metrics since benign traffic is the most common type in the dataset. Considering the high FP rates of conventional signature-based IDS, a ratio of 0.40% is highly encouraging. The current FPR of 0.4% is extremely low and cannot be used as a benchmark. **Fig 10** displays the results of the classification analysis. To get an overall average, a macro-average method first calculates the statistic for each group separately before averaging them. Here the weighted-average F1 score is determined by calculating the mean of all per-class F1 scores while considering each class's support.

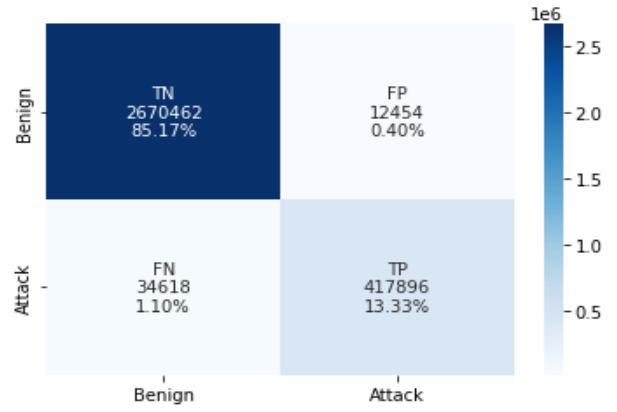

Fig 9. Confusion Matrix - Two-Class Classification

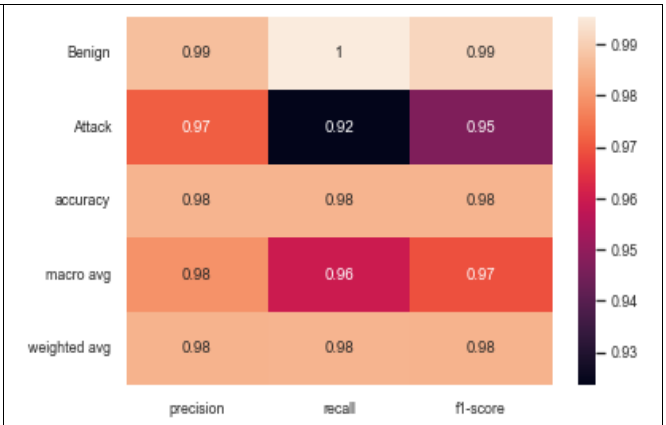

Fig 10. Two-Class Classification Results

FOA-CNN-LSTM model achieves greater accuracy than comparable experiments. For instance, the maximum accuracy reported for binary classification using LSTM on the same dataset CSE-CIC-IDS2018 with a variety of settings is 97.31% (Ferrag et al., 2020) [20]. **Fig 11** shows the ROC Curve and AUC values for normal and malacious traffic, indicating that a ratio of FPR to TPR is greater than threshold limit (FPR=TPR). For a competent classifier, this is a natural outcome. **Fig 12** displays the PR curve and AUC values for benign and attack traffic, demonstrating that both curves lie above the cut-off point. The same is true for the intended outcome of a better classification model.

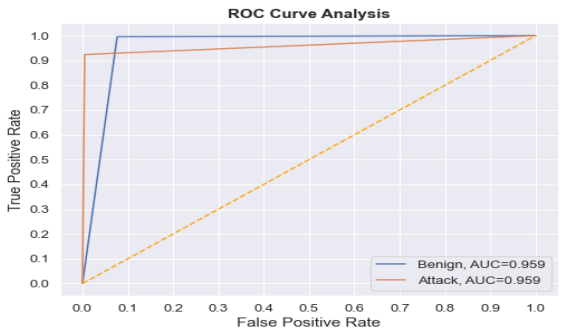

Fig 11. ROC- AUC Curve for Two-Class classifier

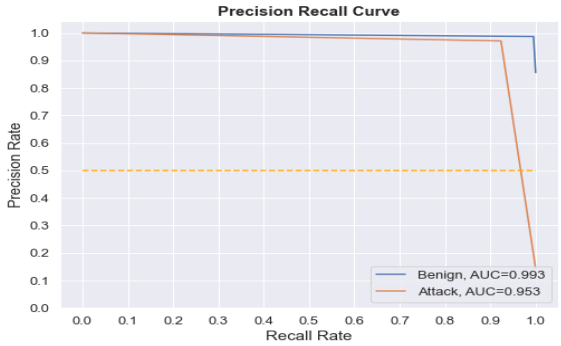

Fig 12. Precision-Recall Curve on Two-Class Classification

The goal of using multiple attack class classification with FOA-CNN-LSTM to make the forecast based on attack classes is explained by continuously increasing the hyper-parameter values based on the lowest prediction error at each iterative level, beginning with level-0 optimization and continuing all the way through level-k optimization, FOA executes multi-class training to develop an optimized CNN-LSTM network.

Table 4 summarizes the results of the study. Overall, the F1 Score, recall, precision, and accuracy are all close to what one would get with a simple yes/no classification. G-Mean and balanced accuracy for class imbalance is deterministic measures much smaller than binary classification. Some minority classes having no Recall value is the primary cause of a null G-Mean. Equally disappointing to those used to binary classification is the fact that the accuracy for balanced categorization is lower as it prioritizes FNs over conventional accuracy. When there are multiple classes to sort, we also compare the FN values for each class to the average FN value.

This outcome can be attributed to subsets of the population with low recall values. **Fig 13** provides the confusion matrix for each one-vs-the-rest category. The most valuable performance indicator is highlighted in dark blue in **Fig 13**. Because the Benign class makes up the bulk of the sample, the True Negative parameter of the malicious traffic outweigh those of the other classes. In **Table 4**, we can see the outcomes of our performance analysis for Multiple Class classifier.

Table 4. Performance Results of Multiple Class classifier

Metrics	Outcomes
Accu	0.9931
Recall	0.9930
Precision	0.985
F1-Score	0.985
G-Mean	0.000
Balanced Accu	0.599

The model cannot learn brute-force attacks, and sql injection threats because their corresponding TP values are 0. These assaults have the distinction of being the least frequent in the sample. The model detects the assaults like brute-force attacks as DDoS or normal. Further, the high False Negative value of Infiltration lowers both recall and accuracy.

Fig 14 displays the outcomes of the classification study. For classification issues involving many classes, the average micro method averages the four possible outcomes (TP, TN, FN, and FP) for each label before arriving at the final classification. When applied to unbalanced data, it produces false results. The overall performance of this model can be more accurately judged by looking at its mean macro value. When figuring out results, it considers an unweighted average of all labels. **Fig 15** displays the ROC-AUC curves for each classification. All classes are displayed over the threshold despite having low precision and recall values. Since TPR and FPR tend to favor the majority class, this condition is misleading for imbalanced multi-class categorization since it emphasizes both Precision and Recall values, the Precision-Recall curve depicted in **Fig 16** for each class yields more realistic results.

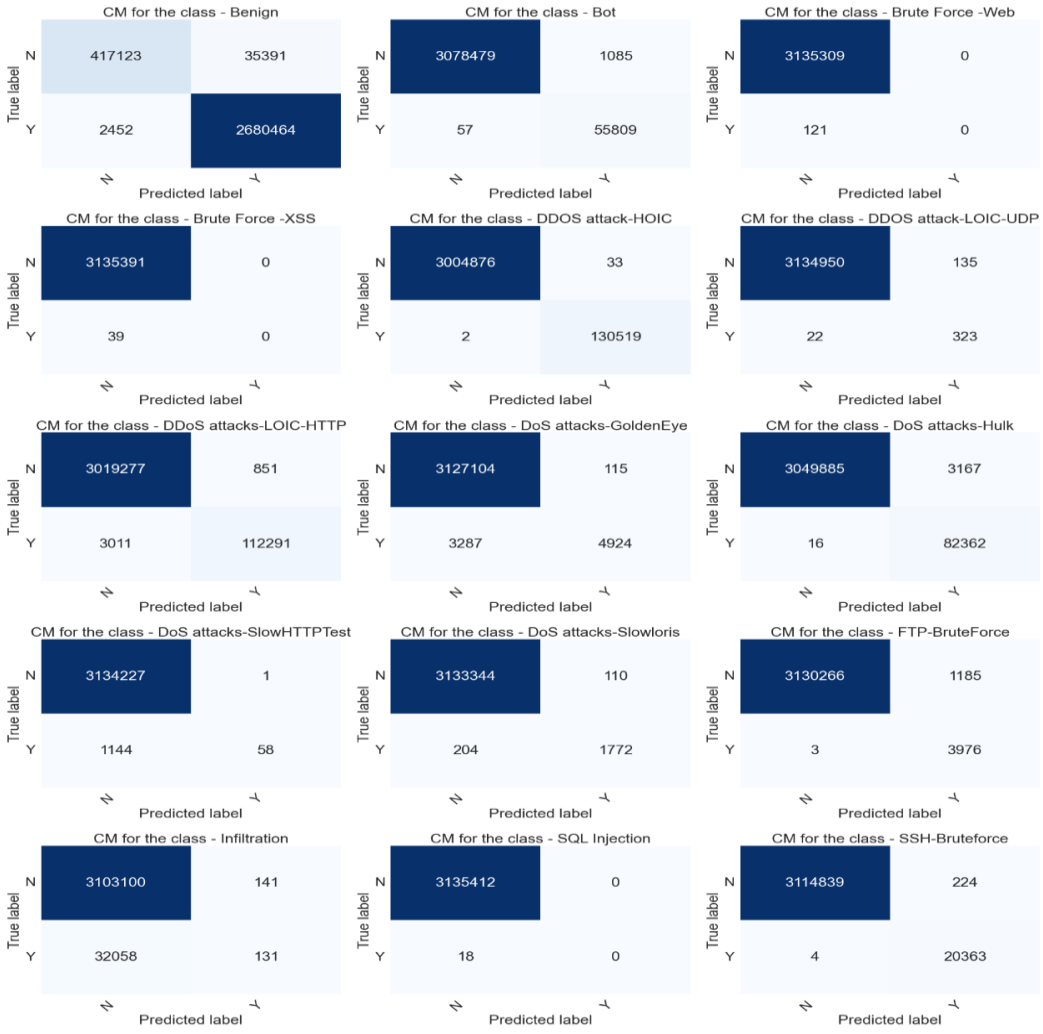

Fig 13. Confusion Matrix - Multi-class Classification

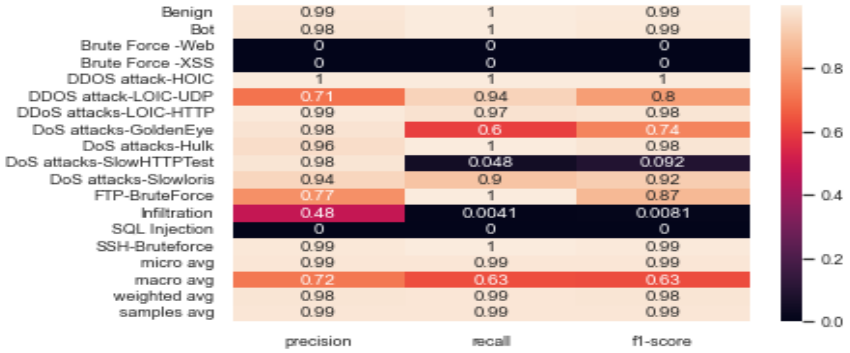

Fig 14. Multiple Attack Class Classifier Report

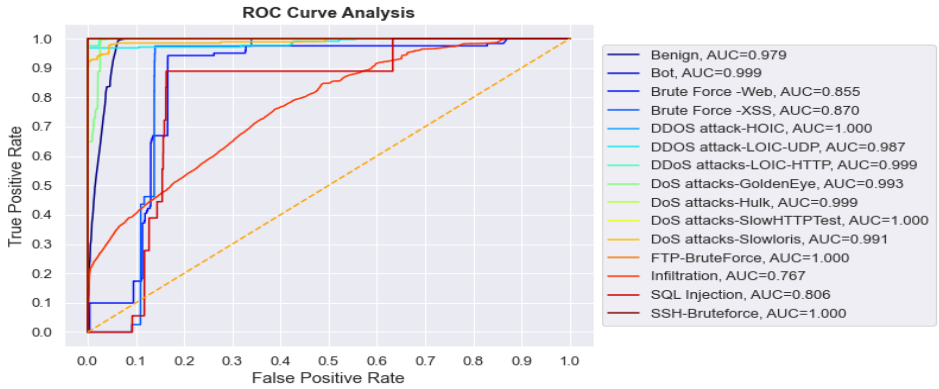

Fig 15. ROC-AUC for Multiclass Classifier

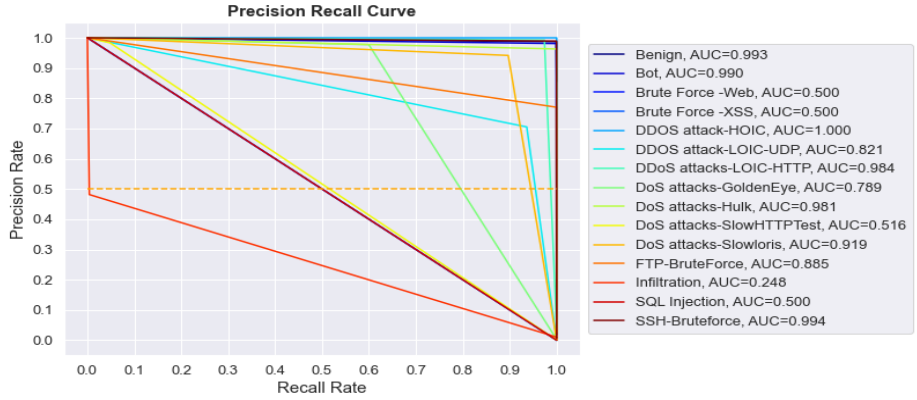

Fig 16. Precision-Recall for Multiclass Classifier

This study uses CNN, LSTM, CNN-LSTM, MLP, Random Forest (RF), SVM, and Naïve Bayes (NB) as comparative experiments. The results of CNN, LSTM, CNN-LSTM, MLP, Random Forest (RF), SVM, and Naïve Bayes will be compared with the experimental results of FOA-CNN-LSTM. In **Table 5**, we can see Comparative analysis of FOA-CNN-LSTM with other models.

Table I. Comparative analysis of FOA-CNN-LSTM Model With Other Model

Model	Accuracy	Precision	Recall	F1 Score
CNN	96.15	98.22	91.2	94.14
MLP	87.25	89.32	87.32	88.12
LSTM	97.11	98.39	90.1	94.1
CNN-LSTM	97.89	98.1	98.2	98.14
SVM	96.4	98.15	98.25	98.24
Naïve bayes	96.29	93.23	93.12	91.98
RF	95.14	91.18	91.31	91.5
FOA-CNN-LSTM Binary Classification	99.01	99.33	98.5	98.5
FOA-CNN-LSTM Multiclass Classification	99.02	99.3	98.5	98.5

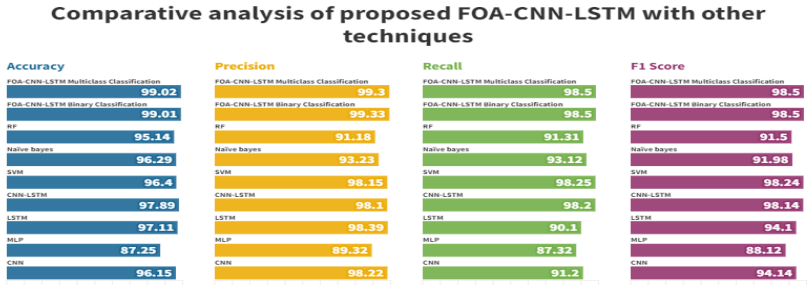

Fig 17. Comparative analysis of Proposed FOA-CNN-LSTM with other techniques.

Fig 17 compares the overall accuracy achieved on the testing dataset by deep learning (DL) techniques trained with the SVM, NB, and RF machine learning methods. Model specifications are tabulated in **Table 1**. For comparison, we get an accuracy of 96.15 percent with the CNN model, 87.25 percent with the MLP, 97.11 percent with the LSTM, 97.89% with the CNN-LSTM, 99.32 % with FOA-CNN-LSTM (Binary Classification) and 99.31 % with FOA-CNN-LSTM (Multi-class Classification). The figure demonstrates that the FOA-CNN-LSTN model achieved the highest level of accuracy while the MLP layer achieved the lowest. **Fig 18** shows different machine learning model comparisons like SVM (Support Vector Machine), which has an accuracy of 96.4%, NB at 96.29%, and RF at 95.14%.

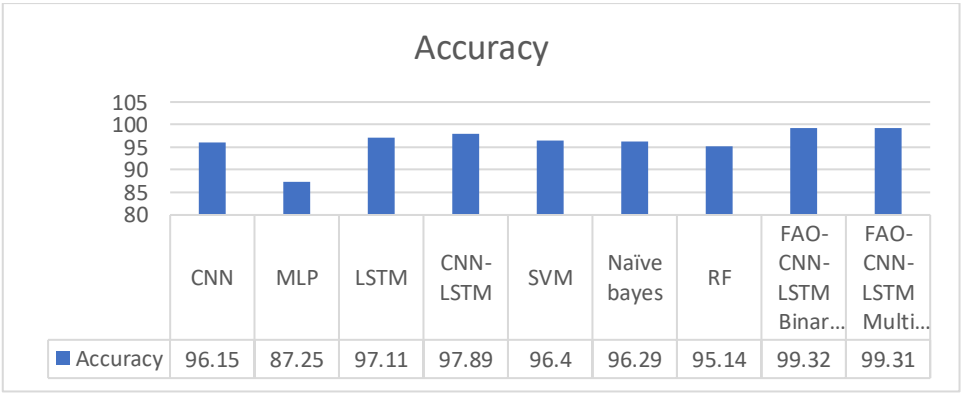

Fig 18. Accuracy Evaluation of Suggested Deep and Machine Learning Techniques.

Precision results from DL and ML approaches are compared in **Fig 19**. The parameters were set the same way as when an accuracy value was obtained. Precision is 98.22% for the CNN model, 89.32% for the MLP model, 98.39% for the LSTM model, 98.1% for the CNN-LSTM, 99.33% for the FOA-CNN-LSTM (Binary Classification) and 99.3% for the FOA-CNN-LSTM (Multi class classification). It is noteworthy that the hybrid model with ensemble and FOA optimization technique is approximately 1.09% more precise than LSMT alone. The diagram demonstrates that SVM achieves 98.15% accuracy, NB achieves 93.23% accuracy, and the random forest achieves 91.18% accuracy. The accuracy achieved by the multi-layer perceptron model is inferior to that achieved by machine learning methods, whereas the accuracy achieved by other models is superior. FOA-CNN-LSTM model has higher accuracy than SVM by about 1.15%.

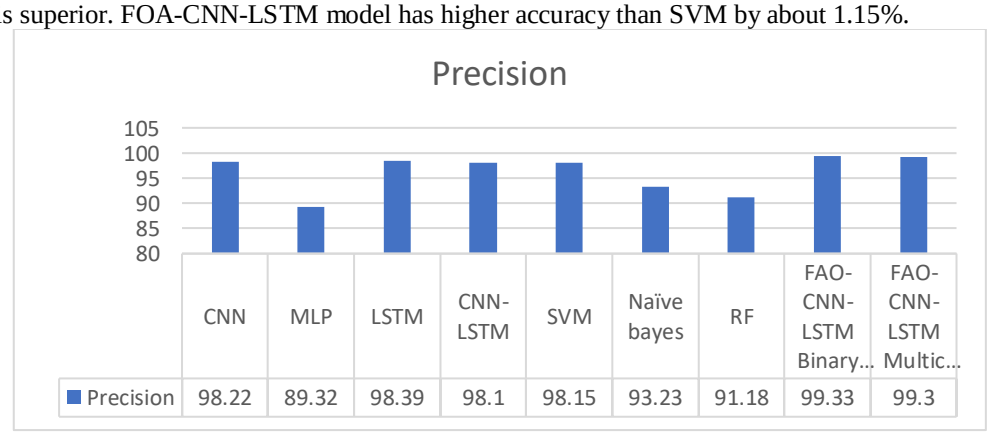

Fig 19. Precision Evaluation of Suggested Deep and Machine Learning Techniques.

Fig 20 summarizes the comparison of recall matrices with CNN is 91.2%, MLP it is 87.32%, LSTM it is 90.1%, and CNN-LSTM it is 98.2%, 98.5% for the FOA-CNN-LSTM (Binary Classification) and 98.5% for the FOA-CNN-LSTM (Multi class classification). FOA-CNN-LSTM has better accuracy and recall than competing models. **Fig 20** shows that

MLP performance remains the worst of all the graphs. From the results, it is safe to say that the FOA-CNN-LSTM model outperforms competing deep learning models and machine learning algorithms in identifying malicious behavior.

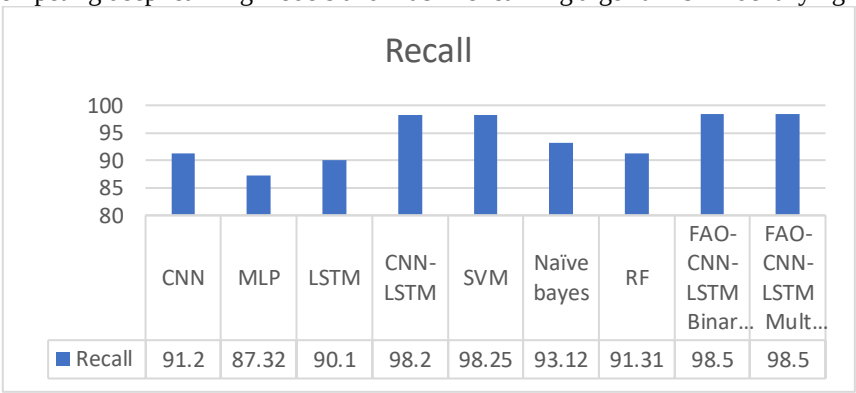

Fig 20. Precision Evaluation of Suggested Deep and Machine Learning Techniques.

Fig 21 summarises the comparison of F1 Scores with CNN is 94.14%, MLP it is 88.12%, LSTM it is 94.1%, and CNN-LSTM it is 98.14%, 98.5% for the FOA-CNN-LSTM (Binary Classification) and 98.5% for the FOA-CNN-LSTM (Multi class classification). When compared to other models, MLP's F1-score of 88.12 percent is once again the lowest. To conclude, the FOA-CNN-LSTM model can acquire a high F1- score of 98.5%, demonstrating exceptionally low FN and FP rates compared to other DL and ML models. The F1-Scores for SVM, NB, and RF, three machine learning models, are 98.24%, 91.98%, and 91.5%. This could explain why Support Vector Machine has been widely utilized as a classification in various uses before the DL methods began to operate in practice.

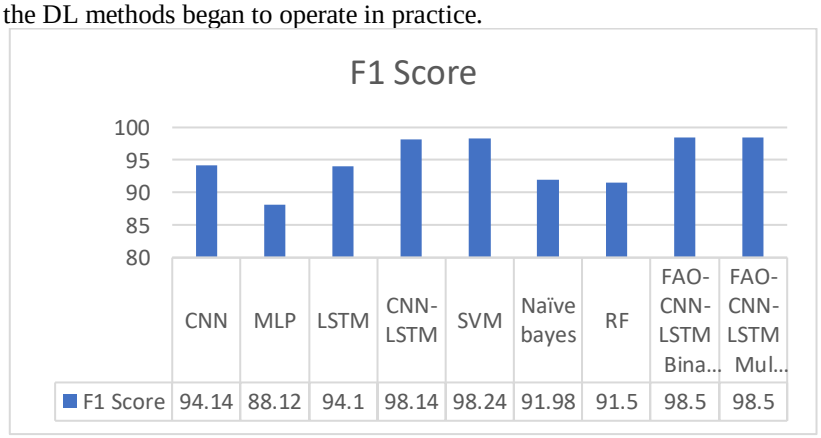

Fig 21. F1-Scores Evaluation of Suggested Deep and Machine Learning Techniques.

V. CONCLUSION

The suggested model used network traffic data to assess the performance of the DL strategy for intrusion detection in this study. The CSE-CIC-IDS2018 dataset, which includes examples applicable to real-world problems, was used to construct a deep-learning solution. Network flow analysis against deep packet inspection for intrusion detection has been used to test the suggested model. The outcomes were quite encouraging, demonstrating that certain attack traffic may be spotted via network flow analysis. Due to its lower resource requirements and ability to account for SSH traffic, the proposed model is particularly well-suited for application in practical settings, as shown by raw traffic analysis. Most internet data is now encrypted. Therefore, doing a deep packet inspection requires in-line analysis. Because of this, network flow analysis becomes more useful in these situations. The model took several routes, each of which made use of unique DL and ML methods. After feature engineering, several deep learning models were developed by picking the most appropriate hyper parameter.

In comparison to the other DL and ML techniques, the hybrid FOA-CNN-LSTM model with an ensemble approach achieves the highest accuracy (99.33%) and F1-Score (98.50%). The suggested deep learning-based approach FOA-CNN-LSTM uses an optimized CNN-LSTM network, which is built by raising the hyper-parametric values at each iteration from level 0 to level k in order to achieve the lowest prediction error. The suggested FOA algorithm 1 automatically adjusts the CNN-LSTM hyperparameters, such as batch size, the hidden layer neurons, the hidden layers, and the epoch. The network is proven accurate when the best possible smell is located using deterministic mapping on top of optimized hidden neurons, optimal weight, and hidden representation. It is determined that the "Attention Mechanism," a more cutting-edge and worthwhile solution, can be used in future work to enhance the solution's performance, in addition to class imbalance solutions.

Data Availability
No data was used to support this study.

Conflicts of Interests
The author(s) declare(s) that they have no conflicts of interest.

Funding
No funding agency is associated with this research.

Competing Interests
There are no competing interests.

References

[1] J. Armin, B. Thompson, D. Ariu, G. Giacinto, F. Roli, and P. Kijewski, "2020 Cybercrime Economic Costs: No Measure No Solution," 2015 10th International Conference on Availability, Reliability and Security, Aug. 2015, doi: 10.1109/ares.2015.56.
[2] J. S. Nye, "Deterrence and Dissuasion in Cyberspace," International Security, vol. 41, no. 3, pp. 44–71, Jan. 2017, doi: 10.1162/isec_a_00266.
[3] N. K. Raja, K. Babu, A. Senthamaraiselvan, and K. Arulanandam, "Routers Sequential Comparing Two Sample Packets for Dropping Worms," International Journal of Computer Network and Information Security, vol. 4, no. 9, pp. 38–46, Aug. 2012, doi: 10.5815/ijcnis.2012.09.05.
[4] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K.-S. Kwak, "An Enhanced Anomaly Detection in Web Traffic Using a Stack of Classifier Ensemble," IEEE Access, vol. 8, pp. 24120–24134, 2020, doi: 10.1109/access.2020.2969428.
[5] Umar, M. A., Zhanfang, C., & Liu, Y. , "A Hybrid Intrusion Detection with Decision Tree for Feature Selection", arXiv preprint, 2020), arXiv:2009.13067.
[6] A. Abdollahi and M. Fathi, "An Intrusion Detection System on Ping of Death Attacks in IoT Networks," Wireless Personal Communications, vol. 112, no. 4, pp. 2057–2070, Jan. 2020, doi: 10.1007/s11277-020-07139-y.
[7] P. Kumar, G. P. Gupta, and R. Tripathi, "Toward Design of an Intelligent Cyber Attack Detection System using Hybrid Feature Reduced Approach for IoT Networks," Arabian Journal for Science and Engineering, vol. 46, no. 4, pp. 3749–3778, Jan. 2021, doi: 10.1007/s13369-020-05181-3.
[8] K. Adhikary, S. Bhushan, S. Kumar, and K. Dutta, "Evaluating the Performance of Various SVM Kernel Functions Based on Basic Features Extracted from KDDCUP'99 Dataset by Random Forest Method for Detecting DDoS Attacks," Wireless Personal Communications, vol. 123, no. 4, pp. 3127–3145, Oct. 2021, doi: 10.1007/s11277-021-09280-8.
[9] M. Li, D. Han, X. Yin, H. Liu, and D. Li, "Design and Implementation of an Anomaly Network Traffic Detection Model Integrating Temporal and Spatial Features," Security and Communication Networks, vol. 2021, pp. 1–15, Aug. 2021, doi: 10.1155/2021/7045823.
[10] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE Access, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/access.2019.2895334.
[11] S. Einy, C. Oz, and Y. D. Navaei, "The Anomaly- and Signature-Based IDS for Network Security Using Hybrid Inference Systems," Mathematical Problems in Engineering, vol. 2021, pp. 1–10, Mar. 2021, doi: 10.1155/2021/6639714.
[12] K. Kim, M. E. Aminanto, and H. C. Tanuwidjaja, "Deep Feature Learning," Network Intrusion Detection using Deep Learning, pp. 47–68, 2018, doi: 10.1007/978-981-13-1444-5_6.
[13] W.-T. Pan, "A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example," Knowledge-Based Systems, vol. 26, pp. 69–74, Feb. 2012, doi: 10.1016/j.knosys.2011.07.001.
[14] A. Das and . P., "An Approach for Identifying Network Intrusion in an Automated Process Control Computer System," International Journal of Electrical and Electronics Research, vol. 10, no. 4, pp. 1219–1224, Dec. 2022, doi: 10.37391/ijeer.100472.
[15] S. Borah, R. Panigrahi, and A. Chakraborty, "An Enhanced Intrusion Detection System Based on Clustering," Progress in Advanced Computing and Intelligent Engineering, pp. 37–45, Dec. 2017, doi: 10.1007/978-981-10-6875-1_5.
[16] D. Baskaya and R. Samet, "DDoS Attacks Detection by Using Machine Learning Methods on Online Systems," 2020 5th International Conference on Computer Science and Engineering (UBMK), Sep. 2020, doi: 10.1109/ubmk50275.2020.9219476.
[17] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," Computers & Security, vol. 88, p. 101645, Jan. 2020, doi: 10.1016/j.cose.2019.101645.
[18] W. Guo and Z. Zhao, "A Novel Hybrid BND-FOA-LSSVM Model for Electricity Price Forecasting," Information, vol. 8, no. 4, p. 120, Sep. 2017, doi: 10.3390/info8040120.
[19] H. Aiqin and W. Yong, "Pressure Model of Control Valve Based on LS-SVM with the Fruit Fly Algorithm," Algorithms, vol. 7, no. 3, pp. 363–375, Jul. 2014, doi: 10.3390/a7030363.
[20] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," Journal of Information Security and Applications, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.