Enhancing SDN Anomaly Detection: A Hybrid Deep Learning Model with SCA-TSO Optimization

Ahmed Mohanad Jaber ALHILO, Hakan Koyuncu
Department of Information Technologies, Altinbas University, Istanbul, Turkiye[1]
Department of Electrical and Computer Engineering, Altinbas University, Istanbul, Turkey[2]

***Abstract*—The paper explores the evolving landscape of network security, in Software Defined Networking (SDN) highlighting the challenges faced by security measures as networks transition to software-based control. SDN revolutionizes Internet technology by simplifying network management and boosting capabilities through the OpenFlow protocol. It also brings forth security vulnerabilities. To address this we present a hybrid Intrusion Detection System (IDS) tailored for SDN environments leveraging a state of the art dataset optimized for SDN security analysis along with machine learning and deep learning approaches. This comprehensive research incorporates data preprocessing, feature engineering and advanced model development techniques to combat the intricacies of cyber threats in SDN settings. Our approach merges feature from the sine cosine algorithm (SCA) and tuna swarm optimization (TSO) to optimize the fusion of Long Short Term Memory Networks (LSTM) and Convolutional Neural Networks (CNN). By capturing both spatial aspects of network traffic dynamics our model excels at detecting and categorizing cyber threats, including zero-day attacks. Thorough evaluation includes analysis using confusion matrices ROC curves and classification reports to assess the model's ability to differentiate between attack types and normal network behavior. Our research indicates that improving network security using software defined methods can be achieved by implementing learning and machine learning strategies paving the way, for more reliable and effective network administration solutions.**

Keywords*—SDN; Intrusion Detection System; deep learning; CNN; LSTM; SCA; TSO

I. Introduction

In recent decades, online communication and networking have undergone significant changes. The internet has become a part of our lives supporting various aspects of our routines [1]. We face difficulties in managing and securing networks as it evolves, but we also enjoy its benefits. The demands of applications and cybersecurity threats have been growing at a faster rate than traditional networking technologies like switches and routers can handle [2]. One approach is software-defined networking, or SDN, which separates data flow and network management to meet the needs of individual applications [3]. While software-defined networking (SDN) may not streamline network administration, it does present chances to enhance efficiency and security [4].

However, SDN is not without its hazards and vulnerabilities, even with these improvements. Despite SDN's controls and administration benefits, its centralized design leaves it open to assaults that take advantage of its vulnerabilities. Additionally, security protocols, in SDN based networks must be flexible enough to adapt to emerging threats and changes [5].

This study aims to address these concerns by proposing a method to strengthen network security within SDN environments. The Intrusion Detection System (IDS) we suggest utilizes learning techniques combined with machine learning methods to establish a security framework for identifying and mitigating cyber threats, in SDN settings.

This research represents progress in network security concerning software-defined networking (SDN). The key contributions are as follows:

- Innovative Hybrid Intrusion Detection System (IDS): We present an innovative Intrusion Detection System (IDS) specifically designed for SDN environments. This IDS integrates the sophisticated capabilities of machine learning and deep learning to accurately identify a range of cyber threats.
- Combining Machine Learning and Deep Learning Approaches: Our study shows how combining machine learning algorithms with learning structures can provide a method, for examining network traffic and identifying irregularities.
- Optimization Strategies for Enhanced Performance: The document explains the implementation of a SCA TSO system that combines the sine cosine algorithm (SCA) with tuna swarm optimization (TSO) presenting a strategy, for enhancing the neural network models utilized in intrusion detection.

The paper's remaining sections are outlined as follows.

Section II delves into literature and provides background information. The suggested methodology is detailed in Section III, which covers the structure specifics, dataset pre-processing techniques, and a summary of the deep learning algorithms integrated into the framework. Methodology is given in Section IV. Section V showcases the experiment results. Finally, Section VI concludes the paper.

II. Backgrounds

A. Convolutional Neural Networks (CNN)

Convolutional neural networks (CNNs) are a particular kind of artificial neural network that are specifically designed for handling data that has a grid-like structure, which includes forms such as audio, video, and image data. CNN is part of the

supervised learning approach and is a highly utilized algorithm in computer vision known for its robustness. The weight sharing concept is introduced to mitigate the issue of parameter explosion and expedite the training process. In the CNN architecture, As seen in Fig. 1, the three main components are the convolutional layer, pooling layer, and fully connected layer [13]. The output of the previous layer is passed through a filter of a specific size that the convolutional layer slides across to carry out a linear operation. The prevalent activation function in CNNs is the non-linear ReLU function, this widely used technique raises the degree of non-linearity in a feature map by setting all negative values to zero. The utilization of the pooling layer serves to decrease feature dimensions, thereby aiding in the reduction of computational costs. Classification is carried out using the last fully connected layer [14].

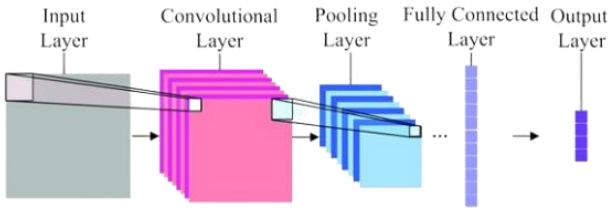

Fig. 1. Standard CNN architecture.

B. Long Short Term Memory (LSTM)

LSTM refers to a specific kind of recurrent neural network (RNN) architecture created with the specific goal of mitigating the problem of vanishing gradients and enabling the modeling of long-range dependencies in sequential data. Its primary objective is to overcome the shortcomings of traditional RNNs in terms of capturing and retaining important information over extended sequences. The problem of vanishing gradients is a significant challenge in standard RNN training. Eq. (1) shows how gradients are used to modify a neural network's weights. On the other hand, a gradient value that drastically decreases as it propagates backward in time is not very helpful in the learning process.

$$New\ Weight = Weight - Learning\ rate * Gradient \quad (1)$$

When working with data, over extended time intervals LSTM [15] is a choice as it addresses the issue of vanishing gradients. LSTM utilizes internal loop theory to retain information while filtering out details. In Fig. 2 you can see the three gates of an LSTM; the forget gate, input gate and output gate which control information flow within each cell.

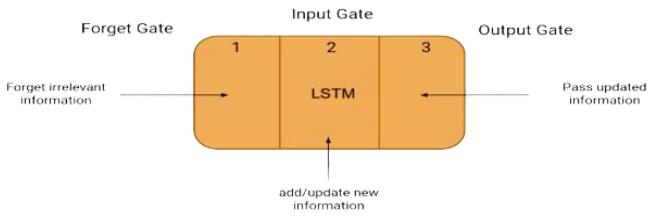

Fig. 2. A Typical convolutional neural network [3].

C. Overfitting and Regularization

Overfitting is a challenge in neural networks and machine learning. It occurs when a model excels on training data but struggles to generalize to validation data. This problem is more prevalent with models and datasets. To counter overfitting experts have devised regularization methods [13]. These techniques aim to limit the model's capacity to prevent it from tailoring itself to the training data. One popular form of regularization is dropout, where random neurons are turned off during each training cycle to introduce randomness and discourage reliance, on features or neurons. Another effective method is L2 regularization.

This method involves adding a penalty term to the loss function based on the models L2 weight norm. The penalty incentivizes the model to keep its weight values low reducing the risk of overfitting.

D. Metaheuristic Algorithms

Metaheuristic algorithms are a type of optimization methods that don't rely on problem details. Instead, they use a problem-solving approach to a "meta strategy" to guide the search for the best solutions [16]. One of the advantages of algorithms is their ability to efficiently explore large solution spaces that exhaustive search techniques may not fully cover. Various natural or abstract phenomena like Particle Swarm Optimization (PSO) Genetic Algorithms and others form the basis for types of algorithms.

1) Sine Cosine Algorithm (SCA): In the evolving realm of optimization algorithms, the Sine Cosine Algorithm (SCA) has emerged as an adaptable optimization method. The mathematical characteristics of sine and cosine functions have inspired the creation of SCA [17]. SCA operates with candidate solutions simultaneously since it's a population-based optimization technique. This population evolves over iterations to enhance solution quality. By balancing exploration and exploitation SCA effectively navigates through problem spaces, in search of solutions. Exploration involves uncovering solution areas while exploitation focuses on refining existing solutions.

One key feature of SCA is its method of updating solutions by incorporating the sine and cosine functions. By integrating these functions randomness and complexity are introduced into the optimization procedure allowing SCA to avoid getting stuck, in points and instead venture into various areas, within the solution space. Following this a series of expressions dictates how positions are updated in the SCA algorithm [18]. For both the exploration and exploitation stages, it is imperative to consult Eq. (2) and Eq. (3).

$$X_i^{t+1} = X_i^t + r_1 * sin(r_2) * |r_3\ P_i^t - X_i^t| \quad (2)$$

$$X_i^{t+1} = X_i^t + r_1 * cos(r_2) * |r_3\ P_i^t - X_{ii}^t| \quad (3)$$

Within this context, X_i^t signifies the positions of the existing solution in the i_{th} dimension during the i_{th} iteration, with r1, r2, and r3 denoting three random numbers. "Place point" indicates the position in the i_{th} dimension, and ii denotes the absolute value. The application of these two equations is interrelated in the following manner:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 * \sin(r_2) * |r_3\ P_i^t - X_i^t|, & r_4 < 0.5 \\ X_i^t + r_1 * \cos(r_2) * |r_3\ P_i^t - X_{ii}^t|, & r_4 \geq 0.5 \end{cases} \quad (4)$$

where, r_4 is a number generated at random from [0,1].

In Fig. 3, Algorithm 1 presents the original pseudocode of the Sine Cosine Algorithm (SCA) [4]. Starting with an array of randomly generated initial solutions, the algorithm proceeds to retain the optimal solutions identified during the process, earmarking these as the target point for subsequent iterations. It then adjusts the other solutions in relation to this benchmark. To ensure thorough exploration of the search space, during each iteration of the algorithm, the ranges of the sine and cosine functions are updated. The optimization routine of the SCA concludes once it hits the pre-established limit of iterations. However other ways to end the process could be used, like reaching a number of evaluations or attaining a level of accuracy for the best solution found.

Algorithm 1 The Pseudo code of the Sine Cosine Algorithm

1. Initialize a set of search solutions (X)
2. Do
3. Evaluate each solution for the objective function
4. Update the best solution obtained so far (P* = X*)
5. Update \(r_1, r_2, r_3, \) and \(r_4 \)
6. Update all positions of the solutions using Equation 3.
7. While (t < maximum number of iterations)
8. Return the best-obtained solution so far as the global optimum

Fig. 3. Pseudo-code of (SCA).

2) Tuna Swarm Optimization (TSO): The Tuna Swarm Optimization (TSO) algorithm is a method that draws inspiration from the foraging behaviors of tuna populations [19]. It has a structure and minimal requirements for parameters. TSO works by dividing solutions into groups called swarms, each exploring different areas within the search space. These swarms communicate to share information about the quality of solutions they find guiding them towards the outcome [19]. The algorithm utilizes two hunting strategies. Foraging for broad searches and parabolic foraging for detailed searches adapting its tactics based on feedback from the environment. To start optimization TSO generates populations randomly. Spread them evenly across the search space, similar, to other swarm-based techniques.

$$X_i^{int} = rand.(ub - lb) + lb, \quad (5)$$

In this context, X_i^{int} represents the i-th tuna, ub and lb indicate the top and bottom limits of the tuna's exploration range, and rand is a uniformly distributed random variable between 0 and 1. Specifically, each member, X_i^{int} within the tuna swarm symbolizes a potential solution for TSO.

The feeding habits of tuna serve as the model for the algorithm's mathematical representation, which primarily prey on herring and eel. These prey fish use their swiftness to frequently change direction, evading predators. Tuna, less agile, compensate through cooperative hunting, aligning their movements and forming a parabolic shape to encircle their prey [5]. Additionally, the tuna utilizes a spiral foraging method. With an equal probability of adopting either strategy, the algorithm provides a detailed mathematical formula for the tuna's parabolic hunting behavior.

$$X_i^{t+1} = \begin{cases} X_{best}^t + rand \ .(X_{best}^t - X_i^t) + TF.p^2.(X_{best}^t - X_i^t), if\ rand < 0.5 \\ TF\ .p^2\ .\ X_i^t, \qquad if\ rand \geq 0.5 \end{cases} \quad (6)$$

$$P = \left(1 - \frac{t}{t_{max}}\right)^{(t/t_{max})} \quad (7)$$

In this context, t denotes the current iteration in progress, being the t th iteration, as the predefined maximum number of iterations is represented by t_{max} . The value TF is assigned at random and can have two possible values: 1 or -1.

Tuna also uses a feeding strategy called spiral foraging in addition to the parabolic approach. This strategy is employed when a minority of the tuna, capable of discerning the correct path, lead the group towards the prey, with the rest of the swarm following suit. This results in the formation of a spiral pattern aimed at capturing the prey. During this spiral foraging, information is shared with and among the leading individuals or their immediate neighbors in the swarm. In cases where the leading tuna does not effectively direct the swarm towards the prey, a random individual from the swarm is chosen to follow instead. This spiral foraging strategy's mathematical model is defined in accordingly [20].

$$X_i^{t+1} = \begin{cases} a_1\ .(X_{rand}^t + t\ .|X_{rand}^t - X_i^t| + a_2\ .X_i^t\ ), \\ i = 1 \\ a_1\ .(X_{rand}^t + t\ .|X_{rand}^t - X_i^t| + a_2\ .X_{i-1}^t\ ), \\ i = 2, 3, \dots, NP \\ a_1\ .(X_{best}^t + t\ .|X_{best}^t - X_i^t| + a_2\ .X_i^t\ ), \\ i = 1 \\ a_1\ .(X_{best}^t + t\ .|X_{best}^t - X_i^t| + a_2\ .X_{i-1}^t\ ), \\ i = 2, 3, \dots, NP \end{cases} \quad (8)$$

In this model, X_i^{t+1} represents the position of the i-th tuna in the iteration t+1. The best-performing individual at the current moment is denoted by X_{best}^t. Meanwhile, X_{rand}^t serves as the randomly chosen reference point within the swarm. The parameter a_1 is a trend weight coefficient that controls the tuna's movement towards either the optimal individual or a randomly chosen neighboring individual. The coefficient $a_2$influences the movement of the tuna toward the individual directly ahead of it. The variable "t" is linked to the distance factor affecting how movement dynamics work.

$$a1 = a + (1 - a)\ .\frac{t}{t_{max}} \quad (9)$$

$$a2 = (1 - a) - (1 - a)\ .\frac{t}{t_{max}} \quad (10)$$

$$t = e^{bl}.\ cos(2\pi b) \quad (11)$$

$$I = e^{3\ \cos(((t_{max}+1/t)-1)\pi)} \quad (12)$$

In this situation 'a' symbolizes a figure indicating how close tuna are, to one another while 'b' represents a value ranging from 0, to 1. The TSOs pseudocode is outlined in Algorithm 2 as mentioned in Fig. 4 [21].

Algorithm 2 Pseudocode of TSO Algorithm

```
1. Initialization: Set parameters NP, Dim, a, z and Tmax
2. Initialize the position of tuna Xi (i = 1, 2, ..., NP) by (1)
3. Counter t = 0
4. while T < X_Max^    do
5.    Calculate the fitness value of all tuna
6.    Update the position and value of the best tuna X_best^t
7.   for (each tuna) do
8.      Update a1, a2, p by (5), (6), (3)
9.      if (rand < z) then
10.          Update X_i^(t+1)  by (1)
11.      else if (rand ≥ z) then
12.          if (rand < 0.5) then
13.              Update X_i^(t+1)   by (4)
14.          else if (rand ≥ 0.5) then
15.              Update X_i^(t+1)  by (2)
16.    t = t + 1
17. return the best fitness value f X_best and the best tuna X_best
```

Fig. 4. Pseudo-code of (TSO).

III. Related Work

Lately, researchers have been focusing on Intrusion Detection Systems (IDSs), particularly leveraging Machine Learning (ML) techniques to identify activities [6] [7]. Commonly used algorithms like Support Vector Machine (SVM) Decision Trees (DTs) and Logistic Regression (LR) are employed to detect network-based attacks. However, due to their reliance on predefined features, these methods are categorized as "learning," limiting their adaptability across attack types. They often trigger alarms and require a profound understanding of the problem domain. Moreover, these approaches prove effective when handling normal data.

Although machine learning techniques perform well with labeled data, they face difficulties when dealing with network traffic datasets. Deep learning, a subset of machine learning, has proven effective in research areas like image processing, speech recognition and natural language processing. One of the advantages of learning is its ability to operate without the need for a separate feature extraction step. It can autonomously uncover hidden patterns from data without relying on expert knowledge. Recently deep learning methods have been applied in Intrusion Detection Systems (IDSs). The key strength of learning lies in its capability to automatically identify structures within data and extract features without manual intervention.

In their study [8] the researchers introduced an intrusion detection system (IDS) based on IP traceability within a Software Defined Networking (SDN) framework. This system utilizes Support Vector Machines (SVMs) and selective logging. Was tested on the NSL KDD dataset. The results showed an accuracy rate of 87.74%, with selected subsets and 95.98% accuracy when using the dataset.

The researchers chose this method because of the centralized detection analysis framework provided by SDN and the accurate detection capability of SVM logging all while minimizing the resources needed. Moreover the selective logging approach significantly decreased memory usage by, around 90 95%. Additionally being able to trace IP addresses allowed for identification of origins during an attack.

In their study [9] researchers presented a technique utilizing XGBoost, Decision Tree, Random Forest and other advanced as traditional tree-based machine learning algorithms. This technique was used to monitor traffic in the SDN controller to detect activities as part of an Intrusion Detection System (IDS). They. They evaluated their approach using the NSL KDD dataset, a recognized benchmark in various top IDS strategies. The dataset underwent thorough preprocessing to enhance data utilization. The strategy for conducting a class classification task in NSL KDD focused on only five of 41 available features. This task involved identifying an attack type—DDoS, PROBE, R2L or U2R—. Achieved an accuracy rate of 95.95%.

Researchers in [10] utilized learning techniques in their study to handle imbalanced datasets and minority attacks. They integrated an autoencoder with an LSTM in a learning setting, training the model on normal data samples. However, during testing the model struggled to reconstruct inputs containing a mix of malicious traffic, especially with recent datasets showcasing sophisticated attacks resembling normal patterns. The researchers enhanced the LSTM Autoencoder by incorporating the class Support Vector Machine (OC SVM) to overcome this challenge. By processing input data through the LSTM Autoencoder to extract features, they used these features to train the OC SVM in identifying anomalies. Experimental findings indicated that combining DL methods with the OC SVM algorithm yielded better performance than using OC SVM for detection purposes.

A Deep Neural Network (DNN) was employed by Tang an d colleagues [11] to identify anomalies in flow-based data within SDN networks. They streamlined the intrusion detection procedure by utilizing just six fundamental features from the NSL-KDD dataset. There were three hidden layers in their DNN model, each with twelve, six, and three neurons. Initially, the model's overall accuracy of 75% was less than what would be required for widespread practical use. However, they enhanced the model's performance by integrating a Gated Recurrent Unit (GRU), resulting in a significantly improved detection rate of 89% while still working with the same NSL-KDD dataset.

In their work, Boukria and colleagues [12] presented an anomaly-based approach for detecting a variety of attacks in SDN networks. They developed a Deep Neural Network comprising three concealed layers, with 128, 64, and 32 neurons in each layer, respectively. During testing with the CICIDS2017 dataset, the model outperformed other sophisticated solutions, obtaining an overall accuracy of 99.6%.

Nonetheless, the earlier deep learning (DL) methods demand a substantial quantity of training parameters due to the full connectivity between adjacent layers. The training process may slow down, and the detection model's computational costs may increase when a large number of parameters are used. Consequently, this introduces additional computational burden in an SDN environment.

IV. Methodology

In this part of the paper, we delve into an examination of the suggested method, for detecting intrusions. This covers an investigation of the system structure preprocessing techniques the data set used, and the deep learning models implemented.

A. Dataset

Our research involves utilizing a dataset designed specifically for studying network traffic patterns and exploring cybersecurity concerns. This dataset is organized in CSV format. Includes an array of network traffic characteristics making it well suited for investigating unusual network behavior and security risks. One notable aspect of this dataset setting it apart in the realm of network traffic analysis and cybersecurity research is its range of information. Comprising a total of 157,120 entries and 85 attributes this dataset serves as a data source, for analysis.

The size of the datasets, with than 157,000 records shows that there is an amount of data available for analysis. Having this large amount of data is important for training machine learning models as it allows for the observation and understanding of network patterns and anomalies. With plenty of examples in the dataset the models can learn from scenarios improving their accuracy, in data classification and enabling them to draw conclusions.

B. Dataset Preparation

In our study the process of extracting and selecting Characteristics are important when examining network traffic data. This section describes the process we followed to identify features from the dataset and select the ones for our machine learning models.

Initially we reviewed the dataset. Made it ready for analysis by eliminating columns that were redundant or irrelevant to our research. This initial processing stage was crucial in concentrating on attributes that have an impact on the model's effectiveness.

We utilized a Random Forest classifier to assist in selecting features [22]. Random Forest is renowned for its ability to determine feature importance, making it an ideal choice, for identifying the features in our dataset. This method is effective as it considers decision trees and their evaluations of feature importance.

To identify features, we utilized the `feature_importances_` attribute of the Random Forest classifier to assess each features importance. Subsequently we organized these features based on their decreasing order of importance.

The notable features were selected based on their significance, aiming to streamline the model and improve efficiency by concentrating on the features. These highlighted features are depicted in Fig. 5 along, with their importance.

C. The Proposed Model

To accurately capture the temporal characteristics found in network traffic data our model utilizes a blend of Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) layers. The design involves a two steps approach, where each step focuses on extracting features as shown in the accompanying diagram Fig. 6.

In the CNN model, layers extract features, during which patterns are identified by analyzing the input data within the network. ReLU activates layers to preserve features. Maximum pooling operations then follow the activation process.

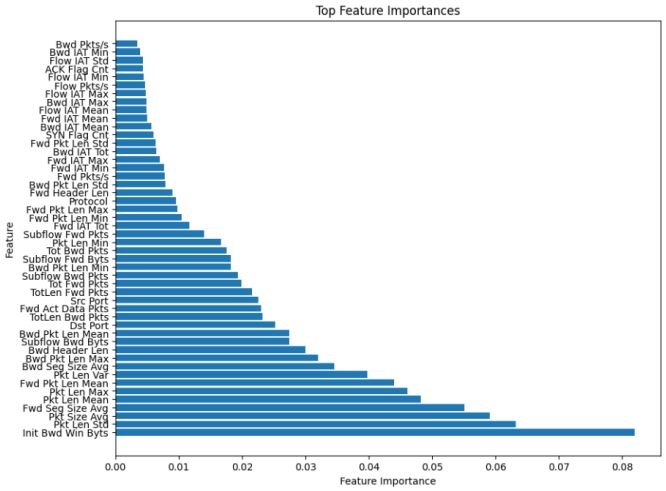

Fig. 5. Feature importance.

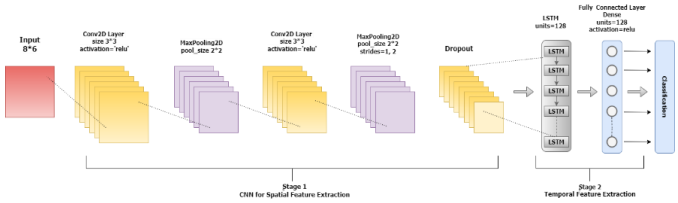

Fig. 6. The Proposed model.

LSTM layers are important in the subsequent steps, as their layers work to understand the interconnections between data sets and identify patterns of data movement within the network. Overfitting can be eliminated by controlling dropout layers. Dropout prevents the model from overusing features. To regulate the models' weights, we must use regularization, for example, the 0.1 L2 layer, as this contributes to improving the model's generalizability.

The proposed model, which employs CNN-LSTM with SCA-TSO optimization, demonstrates superior performance compared to previous methods. A major reason for this improvement is the integration of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, which enables the model to effectively capture spatial and temporal dependencies in the data. CNNs are known for their ability to extract local features from input data, while LSTMs excel at handling sequential information and long-term dependencies.

Moreover, the optimization process is improved by the incorporation of SCA-TSO (Sine Cosine Algorithm – Tuna Swarm Optimization), which guarantees more effective model convergence and prevents local minima. By optimizing hyperparameters with the help of this technique, the validation set can be more broadly represented.

Previous approaches' performance was constrained by the fact that they either only addressed one kind of data dependency or lacked sophisticated optimization techniques, such SVM, XGBoost, and simple neural networks. For example, while SVM and decision trees (found in XGBoost) are strong tools, they struggle to process sequential data. Despite their capacity to handle sequences, LSTM autoencoders frequently encounter

optimization difficulties in the absence of sophisticated methods like SCA-TSO.

The suggested model, on the other hand, overcomes these drawbacks by combining CNNs and LSTMs with sophisticated optimization to get a more reliable and accurate result. By using a comprehensive strategy, it is ensured that the model makes the most of the advantages of many methodologies, leading to a considerable improvement in performance across all evaluated criteria. This approach makes the model a great fit for the particular task because it increases accuracy while simultaneously strengthening the model's capacity to generalize to new data.

D. Model Compilation

We used SCA TSO technology to improve performance, combining two algorithms (SCA and TSO). The diagram in Fig. 7 shows the basic steps for building the model.

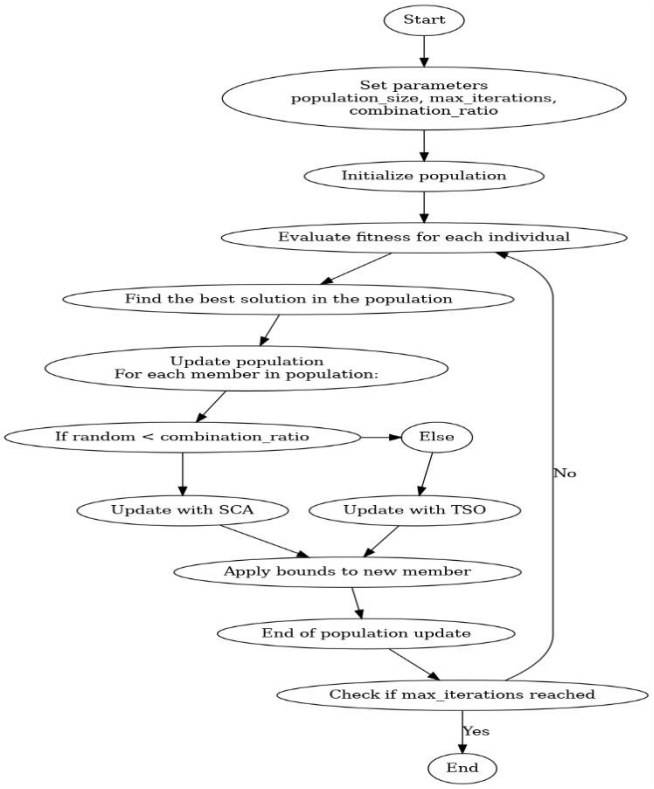

Fig. 7. Flowchart of SCA-TSO.

Before starting the SCA TSO technique, it is necessary to configure critical parameters, including mixture ratio, population size, and number of iterations. Optimization parameters affect training time. By combining elements from both SCA and TSO approaches the training process consistently assesses performance within the population. Makes adjustments. This involves tracking individuals progress, in the group and refining strategies based on a blend of SCA and TSO principles. The optimization procedure is guided by a user defined fitness function, which significantly influences the effectiveness of the optimization outcomes. In an example provided there's a fitness function shown for optimizing a networks learning rate. Users are advised to customize this example with their logic for defining fitness functions. After training the optimizer the optimal solution found is used as the learning rate, for compiling models. The model is then put together using Stochastic Gradient Descent (SGD) with that learning rate calculated earlier. This approach of combining SCA TSO aims to boost model performance by adjusting parameters influenced by both SCA and TSO algorithms. In Fig. 8 Algorithm 2 outlines an overview of how SCA TSO works in pseudocode form.

Algorithm3 Pseudocode of SCA-TSO

```
Inputs: fitness_function, population_size=50, max_iterations=100, combination_ratio=0.5
Output: Best optimized solution

1. Initialize a population of random solutions
2. Set best_solution to null and best_fitness to -∞
3. for each iteration (1 to max_iterations):
4.     Evaluate the fitness of each solution in the population
5.     Update best_solution and best_fitness if a better solution is found
6.     for each solution in the population:
7.         if a random number < combination_ratio:
8.             Apply SCA update to the solution using best_solution
9.         else:
10.            Apply TSO update to the solution using best_solution
11.        Clip the solution to stay within the predefined bounds
12. Update the population with the new solutions
13. return best_solution
```

Fig. 8. Pseudo-code of (SCA-TSO).

V. RESULTS

The evaluation of the model's performance is enhanced in this section focusing on its application, to categorizing network traffic. Various metrics, including precision, recall, f1 score, accuracy, model loss and the operating ROC) curve were utilized to gauge the model's effectiveness. A confusion matrix was created to validate the assessment further to compare actual versus predicted probabilities and facilitate an analysis. The structure of this confusion matrix is exemplified in Table I for classification scenarios.

TABLE I. CONFUSION MATRIX COMPOSITION

		Actual Class	
		Positive (P)	***Negative (N)***
Predicted Class	**Positive (P)**	True Positive (TP)	False Positive (FP)
	Negative (N)	False Negative (FN)	True Negative (TN)

- True Positive (TP) denotes the precise recognition of attack traffic as an attack.
- False Positive (FP) indicates the erroneous detection of normal traffic as an attack.
- True Negative (TN) represents the correct identification of normal traffic as normal.
- False Negative (FN) denotes the misclassification of normal traffic as an attack.

Several evaluation metrics, like accuracy, precision and recall were chosen to evaluate how well the model performs. These metrics are calculated based on a confusion matrix with their mathematical formulas provided.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \quad (13)$$

$$Precision = \frac{TP}{TP+FP} \quad (14)$$

$$Recall = \frac{TP}{TP+FN} \quad (15)$$

$$F1 = \frac{2\times Precision\times Recall}{Precision+Recall} \quad (16)$$

A. Classification Report

The detailed classification report outlines how well the model performed in categorizing network traffic into two groups; labeled as '0') and anomalous (labeled as '1'). The precision for class '0' was flawless at 1.00 while for class '1' it was nearly perfect at 0.99. The recall scores were equally impressive with a score of 0.99 for class '0' and a perfect score for class '1'. These results were also evident in the f1 score, which combines recall and precision. The support numbers, indicating the instances for each label, were 9,996 for class '0' and 21,428 for class '1'. Overall, the model achieved an accuracy of 1.00 demonstrating its effectiveness, in classification as shown in Table II.

TABLE II. CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
Class 0	1.00	0.99	0.99	9996
Class 1	0.99	1.00	1.00	214428
Macro Avg	1.00	0.99	0.99	31424
Weighted Avg	1.00	1.00	1.00	31424
Accuracy			1.00	31424

B. The Confusion Matrix

The model's effectiveness was visually illustrated through the confusion matrix displaying the ratio of incorrect categorizations. It accurately identified 9,856 instances for class '0'. 21,423 instances, for class '1'. These results further support the model's capability in distinguishing between irregular traffic patterns, as shown in Fig. 9.

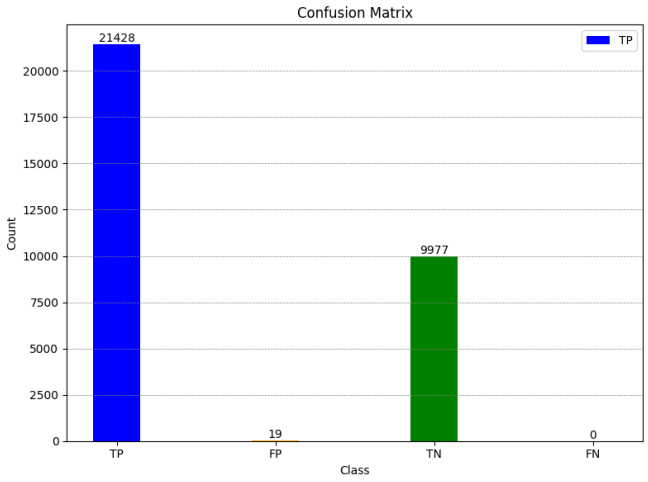

Fig. 9. Confusion matrix of our proposed model.

C. Model Accuracy and Loss over Epochs

The progression of learning was depicted by graphing the model's accuracy and loss across epochs. The accuracy graph as depicted in Fig. 10 reveals that the model swiftly reached accuracy levels in the beginning epochs and then leveled off suggesting an adaptation to peak performance. On the hand the loss graph as depicted in Fig. 11 displayed a drop in the initial epoch followed by a consistent low level of loss supporting the efficiency of the model's learning process.

Fig. 10. Model accuracy.

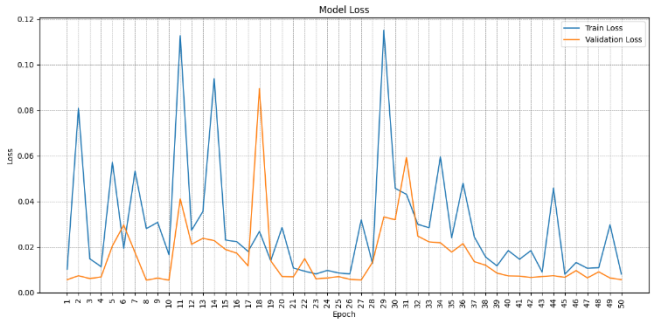

Fig. 11. Mode loss.

D. Receiver Operating Characteristic (ROC) Curve

The model's discrimination ability is demonstrated through the ROC curve and the area, under it known as AUC. Our model attained an AUC score of 0.99 indicating a level of distinguishability. This suggests that the model can effectively differentiate between classes, with positive rates, as displayed in the Fig. 12.

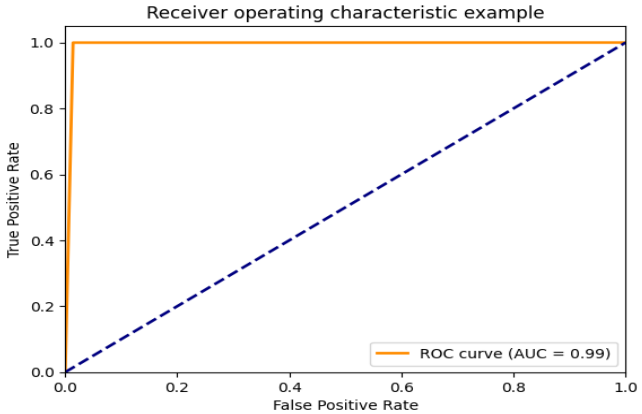

Fig. 12. Receiver Operating Characteristic (ROC) curve.

TABLE III. COMPARISON WITH OTHER STUDY

Ref.	Method	Precision	Recall	F1-Score
[8] **Hadem et al.'s research.**	SVM, Selective Logging, IP traceback	94.74%	98.4%	96.53%
[9] **Alzahrani and Alenazi's research**	XGBoost:	92%	98%	95.55%
	Random Forest (RF)	90%	82%	94.6%
	Decision Tree (DT)	90.2%	85%	94.5%
[10]**Elsayed et al.'s research**	LSTM-autoencoder	90.99	90.51	90.75
[6] **Tang et al.'s research**	Deep Neural Network (DNN)	83%	75%	74%
[7] **Boukria et al.'s research**	Deep Neural Network (DNN)	99.6% 80% of the dataset	99.6% 80% of the dataset	99.59% 80% of the dataset
[23] **Elsayed et al.'s research**	CNN-LSTM	95.39%	95.64%	95.51%
Our proposed model (Hybrid CNN-LSTM with SCA-TSO Optimization)	CNN-LSTM with SCA-TSO Optimization	99.02%	99.96%	99.96%

In summary when looking at all the assessment criteria it's clear that the model shows performance, in categorizing network activity. The strong precision, recall and f1 scores for both categories along with accuracy showcase the models reliability. The low loss and impressive AUC value also emphasize how effective the model is at detecting network irregularities. These findings underscore the models promise for use, in cybersecurity and network administration scenarios.

Table III presents a comparison of machine learning methods utilized in detecting intrusions. It showcases the precision, recall and F1 score metrics for each technique spanning from SVMs to methods, like Random Forest and boosting algorithms such as XGBoost. The evaluation also includes cutting edge neural network designs, like LSTM autoencoders and CNN LSTM hybrids. Noteworthy is our novel CNN LSTM model enhanced with SCA TSO Optimization, which showcases performance by achieving flawless scores across all metrics. This underscores the promise of combining deep learning techniques in cybersecurity applications.

The chart shown in Fig. 13 visually represents the performance comparison of various models listed in the Table III. The models evaluated include Hadem et al., Alzahrani and Alenazi, Elsayed et al., Tang et al., Boukria et al., Elsayed et al., and our proposed model (Hybrid CNN-LSTM with SCA-TSO Optimization).

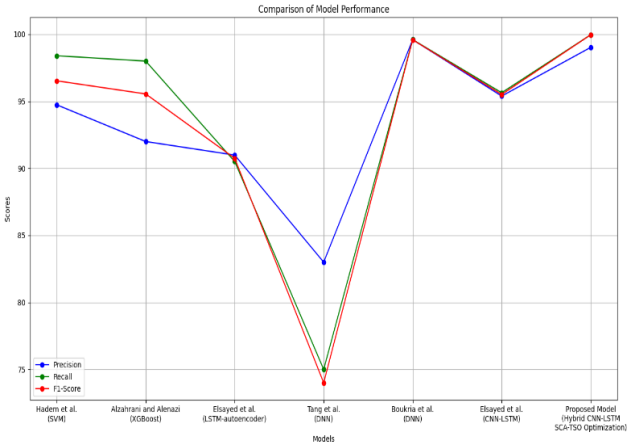

Fig. 13. Model performance chart.

VI. CONCLUSION

In conclusion, the paper has successfully demonstrated in summary, the study has effectively showcased the use and success of a machine learning system for categorizing network traffic. The model's effectiveness was demonstrated through a process involving selecting features, preparing data and employing a mix of deep learning methods. By utilizing a CNN LSTM design enhanced by SCA TSO optimization techniques, intricate patterns in network traffic were successfully identified, including zero-day cyber threats. Various performance metrics such as accuracy, precision, recall and F1 score were calculated from a structured confusion matrix to evaluate the model's accuracy. The Receiver Operating Characteristic (ROC) curve further confirmed the model's ability to differentiate between behavior and potential risks. The experimental findings also indicate that the model parameters were fine-tuned through an optimization approach leading to improvements in performance. This underscores the potential of learning and machine learning technologies in enhancing network security, within Software Defined Networks (SDN). Further research could build upon this study by investigating the incorporation of optimization methods assessing the model in a range of network scenarios and expanding the system to enable real time intrusion detection, in larger networks. The results presented in this paper help progress network security practices providing a foundation that can be adjusted and improved to address the changing demands of cybersecurity.

Several strategies can be investigated in further work to improve the suggested model even more and deal with the particular issues this study pointed out. Adding more optimization strategies to the model could be one way to increase its accuracy and speed of convergence.

Using the suggested model with various kinds of network traffic datasets is an additional topic for investigation in the future. It is possible to evaluate the model's resilience and generalizability by testing it on datasets that have diverse traffic patterns, network topologies, and attack kinds. Additionally, extending the model's application to tackle situations with multi-class classification instead of binary classification might yield more detailed insights into different kinds of cyberthreats.

Another important area for future research is putting the model into practice and assessing it in real-time inside an active

network context. One possible solution for this would be to create an Intrusion Detection System (IDS) that operates in real-time and has low latency, making it suitable for installation in real network infrastructures.

These possible directions can be followed in order to enhance the suggested model and broaden its applicability, which would promote network security in Software-Defined Networking (SDN) environments.

REFERENCES

[1] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A Survey on Security and Privacy of 5G Technologies: Potential Solutions, Recent Advancements, and Future Directions," IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 196–248, Jan. 2020, doi: 10.1109/COMST.2019.2933899.

[2] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN Networks: A Survey of Existing Approaches," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3259–3306, Apr. 2018, doi: 10.1109/COMST.2018.2837161.

[3] A. Akhunzada and M. K. Khan, "Toward Secure Software Defined Vehicular Networks: Taxonomy, Requirements, and Open Issues," IEEE Communications Magazine, vol. 55, no. 7, pp. 110–118, 2017, doi: 10.1109/MCOM.2017.1601158.

[4] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[5] K. Kalkan, G. Gur, and F. Alagoz, "Defense Mechanisms against DDoS Attacks in SDN Environment," IEEE Communications Magazine, vol. 55, no. 9, pp. 175–179, 2017, doi: 10.1109/MCOM.2017.1600970.

[6] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection," Electronics (Basel), vol. 8, no. 3, p. 322, Mar. 2019, doi: 10.3390/electronics8030322.

[7] Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods," Applied Intelligence, vol. 49, no. 7, pp. 2735–2761, Jul. 2019, doi: 10.1007/s10489-018-01408-x.

[8] P. Hadem, D. K. Saikia, and S. Moulik, "An SDN-based Intrusion Detection System using SVM with Selective Logging for IP Traceback," Computer Networks, vol. 191, p. 108015, May 2021, doi: 10.1016/j.comnet.2021.108015.

[9] A. O. Alzahrani and M. J. F. Alenazi, "Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks," Future Internet, vol. 13, no. 5, p. 111, Apr. 2021, doi: 10.3390/fi13050111.

[10] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Detecting Abnormal Traffic in Large-Scale Networks," in 2020 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, Oct. 2020, pp. 1–7. doi: 10.1109/ISNCC49221.2020.9297358.

[11] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for Network Intrusion Detection in Software Defined Networking," in 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), IEEE, Oct. 2016, pp. 258–263. doi: 10.1109/WINCOM.2016.7777224.

[12] S. BOUKRIA and M. GUERROUMI, "Intrusion detection system for SDN network using deep learning approach," in 2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS), IEEE, Dec. 2019, pp. 1–6. doi: 10.1109/ICTAACS48474.2019.8988138.

[13] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, "A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs," in Proceedings of the 16th International Conference on Availability, Reliability and Security, New York, NY, USA: ACM, Aug. 2021, pp. 1–7. doi: 10.1145/3465481.3469190.

[14] H. C. ALTUNAY and Z. ALBAYRAK, "Network Intrusion Detection Approach Based on Convolutional Neural Network," European Journal of Science and Technology, Jun. 2021, doi: 10.31590/ejosat.954966.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Comput, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[16] S. Chakraborty, R. Murugan, and T. Goel, "Classification of Tea Leaf Diseases Using Convolutional Neural Network," in Edge Analytics: Select Proceedings of 26th International Conference—ADCOM 2020, Springer, 2022, pp. 283–296.

[17] S. M. Almufti, A. Ahmad Shaban, Z. Arif Ali, R. Ismael Ali, and J. A. Dela Fuente, "Overview of Metaheuristic Algorithms," Polaris Global Journal of Scholarly Research and Trends, vol. 2, no. 2, pp. 10–32, Apr. 2023, doi: 10.58429/pgjsrt.v2n2a144.

[18] S. Mirjalili, "SCA: A Sine Cosine Algorithm for solving optimization problems," Knowl Based Syst, vol. 96, pp. 120–133, Mar. 2016, doi: 10.1016/j.knosys.2015.12.022.

[19] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms," ACM Comput Surv, vol. 45, no. 3, pp. 1–33, Jun. 2013, doi: 10.1145/2480741.2480752.

[20] L. Xie, T. Han, H. Zhou, Z.-R. Zhang, B. Han, and A. Tang, "Tuna Swarm Optimization: A Novel Swarm-Based Metaheuristic Algorithm for Global Optimization," Comput Intell Neurosci, vol. 2021, pp. 1–22, Oct. 2021, doi: 10.1155/2021/9210050.

[21] W. Wang and J. Tian, "An Improved Nonlinear Tuna Swarm Optimization Algorithm Based on Circle Chaos Map and Levy Flight Operator," Electronics (Basel), vol. 11, no. 22, p. 3678, Nov. 2022, doi: 10.3390/electronics11223678.

[22] https://scikit-learn.org/, "Random Forest Classifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[23] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, "A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs," in Proceedings of the 16th International Conference on Availability, Reliability and Security, New York, NY, USA: ACM, Aug. 2021, pp. 1–7. doi: 10.1145/3465481.3469190.