SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I9P114

Volume 12 Issue 9, 162-177, September 2025

Original Article

OHNN: An Optimized Hybrid Neural Network for Intrusion Detection Systems Using TLBO and CTO

Shourya Shukla[1], Ajay Singh Raghuvanshi[1], Saikat Majumder[1]

[1]*Department of Electronics and Communication Engineering, National Institute of Technology, Raipur, Chhattisgarh, India.*

[1]*Corresponding Author : sshukla.phd2019.ece@nitrr.ac.in*

Received: 14 July 2025 Revised: 16 August 2025 Accepted: 15 September 2025 Published: 29 September 2025

Abstract *- In recent years, cybersecurity has become a hot topic for exploration among researchers. The anomaly-based Detection methods have been widely used for early detection and mitigation of Intrusions. As the hardware is evolving, there is a need for exploration of new features. This is made possible through novel deep learning models. In this paper, an automated construction of various hybrid deep learning models is performed through a twin optimization algorithm. The Teaching Learning Based Optimization and Class Topper Optimization are used to generate new deep learning structures. The model parameters, like the Number of Hidden layers, the Optimizer and the Learning Rate, are controlled by TLBO. CTO was used to select different internal parameters such as Types of layers, Number of nodes per layer, the activation function and initializers. The proposed model was used to explore several deep learning models to train and test on the NSL-KDD dataset. Binary and Multiclass classifiers were explored for different neural network architectures based on TLBO and CTO parameters. The explored models showed comparable accuracy with respect to existing detection models.*

Keywords *- Teaching Learning Based Optimization, Class Topper Optimization, Hybrid Neural Network, Intrusion Detection System, Optimization.*

1. Introduction

Artificial Intelligence models such as Machine learning and Deep learning models have become one of the most popular assets for prediction and forecasting of unknowns in many fields such as healthcare, agriculture, defense, security and many other fields. One such emerging field is the detection of malicious activities in a network. The advancement in the interconnection of devices without human involvement has left the communication nodes under the threat of an attack. These cyber threats are known as Intrusions.

These Intrusions are deliberate, malicious activities that may or may not disrupt or destroy network resources. An Intrusion can be active or passive in nature. An active attack, such as a Denial of Service (DoS) attack, can deplete bandwidth or other resources, such as power, in an energy-constrained environment. On the other hand, passive attacks, such as probe attacks, do not directly hamper network activities but remain dormant and search for vulnerabilities in the system. Passive attacks have proven to be more damaging to a network as they are very difficult to detect.

An Intrusion Detection System (IDS) is a monitoring framework used in a computer or a network traffic analyzer to detect malicious activities [1]. IDS, nowadays, is implemented not only on computers, but also on Cloud services, VANETs, MANETs and other Wireless Sensor Networks (WSN) [2].

The IDS can be of two types based on their detection strategies: Signature-based IDS and Anomaly-based IDS. The signature-based IDS detects known intrusive activities very fast, but fails to detect Zero-day attacks. A Zero-day attack is well detected by a Based IDS, where normal data is learnt and any deviation from normal is said to be malicious in nature. Anomaly-based IDS uses Stochastic, probabilistic, or Artificial intelligence-based learning techniques for training the models on the data.

Different Deep Learning algorithms with novel structures are employed to explore more ways to detect the intrusive activities in an efficient manner. The deep learning model has been evolving with the advancement in processors and Graphics Processing Units (GPUs). Different architectures of deep learning models, such as autoencoders, hybrid models, transfer learning models, etc., are employed for black-box feature extraction.

Deep Learning algorithms are structure-oriented models in which hidden layers are arranged in different possible orders to extract desirable features. There are many possibilities of the Deep Learning architectures resulting in

different feature extraction methods. These possibilities include structure as well as hyperparameter tuning. The number of implementable structures creates a problem of non-polynomial order. Therefore, there is a high need of optimizing the structure of the proposed architecture. Metaheuristic optimization algorithms, such as bio-inspired, human behavior, or physics-inspired techniques, have been utilized to automate these deep structures. This not only improves the accuracy by exploration of new architectures, but also mitigates the human cognitive bias for some specific structures.

With the intrusion and attackers finding new strategies, IDS have to evolve at a faster rate. Novel Features have to be determined. In this paper, these advancements are explored. The hybrid model architectures are explored by using two human-inspired optimisation algorithms, namely, Teaching Learning Based Optimization and Class Topper Optimization, to explore new Optimized Hybrid Neural Network (OHNN) architectures. The combined optimization algorithms mimic the knowledge transfer from teacher to student, and competition among students encourages them to learn more.

The paper's structure is given as follows. Section I briefly introduces the ongoing issues with intrusions and deep learning structures. Section II deals with the Literature Survey and explains the existing algorithms. Methodology is presented in Section III. Section IV gives information about the Experimental Work carried out, and Section V includes the Conclusion of the paper along with the future scope.

2. Related Work

In this section, a rationale survey of the SOTA research in Intrusion Detection Systems has been presented.

Ajawan in [3] proposed a deep learning architecture for a real-time Intrusion Detection System. A four-layer fully connected network had been proposed. The architecture was deployed on IoT devices for the detection of Blackhole, Distributed Denial of Services, Sinkhole and Workhole attacks. Network Emulators have been employed to connect the victim machine to the network virtually. Moreover, the author was able to achieve a high detection accuracy of 93.74%.

Qazi et al. [4] proposed a hybrid deep neural network for network-based IDS. A convolutional recurrent network was utilized to mitigate any malicious packets from the network. A collection of local features was performed by a convolutional neural network layer, whereas the recurrent layer was responsible for feature extraction. The CICIDS2018 dataset was used to train the hybrid architecture. The architecture consisted of two convolutional layers followed by a recurrent network. Then, a dense layer along with a flattened layer was applied to produce output for classification. A high accuracy of 98.90% was achieved by the authors.

Hossain and Islam in [5] proposed an ensemble machine learning algorithm for the detection of intrusions. Ensemble-based bagging algorithms, such as Random Forest (RF). Meanwhile, boosting-based algorithms like Gradient Boosting, Adaboost, XGBoost, and Simple Stacking classifiers were used for detection purposes. Feature extraction was performed on various datasets using correlation analysis. Furthermore, Principal Component Analysis (PCA) was applied to reduce the feature dimension. The authors ensured a very high accuracy of 99% for more than 10 datasets, including NSL-KDD and UNSW-NB15 datasets.

Bhavsar et al. [6] proposed a Pearson Correlation Coefficient-based Convolutional Neural Network. The Pearson correlation coefficient was used for feature extraction. NSL-KDD, which is a prominent dataset, was employed for training. The latest datasets, CICIDS-2017 and IOTID20, were utilised to train the convolutional neural network with Pearson correlation coefficient-based features. A three-layer CNN was used for classification and achieved 99% accuracy.

Wu et al. [7] proposed a transformer-based intrusion detection system. The authors balanced the dimensionality reduction and feature retention trade-off in the work. The structure consisted of six encoder layers, each made of a self-attention network. The class imbalance was dealt with using the SMOTE algorithm for
the creation of synthetic data. 98.58% accuracy was achieved by the anomaly-based detection method.

Kasongo in [8] proposed various feedback neural network architectures. An XGBoost-based feature selection algorithm was employed with feedback networks like RNN. Moreover, GRU and LSTM-based networks were also inferred. Moreover, authors inferred that increased feature space complexity leads to decreased testing accuracy. A three-layer architecture with varying hidden nodes from 15 to 210 was compared.

Korium et al in [9] proposed a model with a z-score for normalization of the features. Regression models were employed for the complexity reduction, and various machine learning algorithms were used as classifiers. The SMOTE algorithm was applied to the synthetic data along with the modified nearest neighbor technique. HyperOpt was used to parameter-tune the IDS for vehicle intrusions.

Musleh et al in [10] proposed a transfer learning based method for IDS. DenseNet and VGG-16 models were used for feature extraction, and machine learning algorithms, such as RF and K-NN, were used for the mitigation of malicious

packets in the network. An autocolor correlogram was applied to the processed data using the proposed method. 98.3% accuracy was attained by the model with fine KNN.

The use of hybrid structures has enabled researchers to explore more features in the deep learning algorithms. Altunay et al suggested a CNN-LSTM-based hybrid model. A two-layer CNN and a three-layer LSTM architecture were proposed [11]. The suggested model achieved 93.84% accuracy.

Logeswari et al. [12] proposed a hybrid feature selection algorithm with Light Gradient Boosting Machine for Intrusion Detection Systems. Correlation-based feature extraction was proposed by the authors. In particular, Pearson's Correlation Coefficient was employed for feature extraction in software-defined radio. Redundant features were eliminated using Random Forest with recursive feature elimination.

Metaheuristic optimization algorithms, which are teaching learning based optimization, have gained much popularity among researchers. Kaushik et al in [13] proposed a teaching learning based optimized intrusion detection system to overcome communication overhead. The author was able to outperform the bat optimization algorithm and the genetic algorithm by a large margin on the UNSW-NB15 dataset. A random forest classifier was employed to predict malicious packets in the network.

Fraihat et al. [14] proposed a network-based intrusion detection system in an IoT NetFlow-based environment. An arithmetic optimization algorithm was employed for feature selection. The proposed algorithm was able to reduce the feature set from 43 features to 7 features. Ensemble tree-based classifiers were employed for classification purposes. The authors suggested two-dimensional redundancy removal to achieve better accuracy. An objective function to minimize the loss as well as minimizer the size of selected features is incorporated.

Alzaqebah et al in [15] proposed a modified grey wolf optimization technique by combining filter- and wrapper-based approaches. This was performed to select important features and discard other extracted features. Most relevant features were selected based on information gain. The extreme learning machine technique was employed for classification purposes, which is a one-layer feed-forward neural network. A fitness function with minimization of false positives and false negatives, and reduced features was employed.

Kilichev et al in [16] proposed a CNN model Optimized using genetic algorithm and particle swarm optimization. GA and PSO were used to explore and exploit the features at the same time through hyper-parameter optimization. Network structure, learning and optimization, as well as the regularization effect, were taken into consideration. Constrained optimization with hyper-parameter tuning was applied for the creation of the CNN layer.

Kunang et al in [17] proposed a similar approach by hyperparameter tuning using an optimization algorithm. An autoencoder, along with a deep neural network, was employed to detect intrusions. Parameter tuning was achieved by utilizing grid search and random search techniques. The algorithm Optimized hyperparameters using metaheuristic algorithms, which proved to be more efficient than built-in hyperparameter optimizers. These parameters were used to generate new deep learning architectures. However, the type of structures was confined to autoencoder and fully connected network layers only.

Imran et al in [18] proposed a cuckoo search optimization algorithm-based intrusion detection system. Errors in predicting the patterns, such as MAE, MSE and RMSE, were minimized to predict the intrusions in the NSL-KDD dataset.

Latif et al in [19] proposed a genetic algorithm-based deep learning framework. Convolutional neural network, genetic algorithm and bootstrap aggregation transfer ensemble techniques were employed by the authors. Signals or instances were converted to images, and a CNN was applied along with GA for fine-tuning. Bootstrapping ensembles were used to make the system robust.

Gupta et al in [20] proposed a modified twin optimization-based deep learning method for the detection and mitigation of intrusions. A hybrid chicken swarm and genetic algorithm were employed for feature selection. The mini-batch K-means clustering algorithm was employed for dimensionality reduction. The Levy flight, which is a feature of the crow optimization algorithm, was used in the chicken swarm optimization for better exploration of the search space.

Khafaga et al suggested a voting classifier along with metaheuristic optimization of whale optimization algorithm guided by dipper throated optimizer [21]. Class imbalance was dealt with using the Synthetic Minority Oversampling Technique (SMOTE). The RPL-NIDS17 dataset was generated by the NetSim simulator to train and test the model.

Based on the literature survey, the following shortcomings were found in the existing methodologies:

- Authors in [11] adapted a hybrid Neural Network with CNN and LSTM layers. However, the human cognitive bias led to the combination of all CNN and LSTM layers. The architecture lacked exploration of different arrangements of these layers for better feature extraction strategies.
- Tuned Neural Network architectures have been

suggested by [17, 19]. However, the methods either provided an optimized hyperparameter tuning method or employed only one type of hidden layer.

- Authors in [16] employed a combined optimization technique for an optimization and regularization-based CNN model. However, the architecture had a fixed number of CNN hidden layers with variable dense layers. Moreover, the architectures lacked exploration on the basis of the number of hidden layers as well as the type of layered architectures.

Incorporating the gaps found in the existing methods, the proposed work proposes the following novel approaches:

- This paper proposes a twin optimization based on TLBO and CTO for exploring novel deep learning structures.
- TLBO controls the structure of the Neural Network, such as the number of hidden layers, optimizer and learning rate.
- CTO controls the hyperparameter of each layer individually, such as the type of hidden layers, number of filters, initializers and activation function.
- The explored architectures were used to identify the malicious packets and detect the type of intrusion using binary and multiclass classifications on the NSL-KDD dataset.
- The proposed paper tries to eliminate the human bias while creating a deep learning architecture using metaheuristic optimization techniques, making not only the learning of the artificial intelligence algorithms automated, but also the architecture of the models automated.

3. Methodology

Machine Learning and Deep Learning models have found many applications in the field of decision making, event detection and other classification and forecasting problems. Deep learning has enabled researchers to implement most of the feature functions using weight approximation. A certain drawback of the deep learning models is the dependency of the model on its architecture. Human cognitive ability and bias have highly confined the capabilities of Neural Networks. In this paper, the authors have employed a twin optimization algorithm for the exploration of new neural network architectures. The Teaching Learning Based Optimization (TLBO) and Class Topper Optimization (CTO) are employed for exploration purposes. The TLBO is responsible for the exploration of new neural network structures. Whereas the CTO is responsible for handling each layer and its hyperparameters. The methodology proposed in this paper is as follows:

3.1. Initialization

The TLBO algorithm controls the outer structure of the hybrid Neural Network, whereas internal parameters are controlled by CTO. The number of layers, the optimization technique used in back-propagation and the learning rate are controlled by TLBO. The type of hidden layer, number of filters or nodes, initialization function and activation functions are controlled by the CTO for each layer separately.

The metaheuristic approaches are initialized by an initial population randomly generated as shown in equation 1.

$$pop = \begin{bmatrix} L_1^i & L_2^i & L_3^i & Opt_1^i & Opt_2^i & Opt_3^i & lr_1^i & lr_2^i \\ LT_{1,1}^i & LT_{1,2}^i & LT_{1,3}^i & LT_{1,4}^i & LT_{1,5}^i & LT_{1,6}^i & LT_{1,7}^i & LT_{1,8}^i \\ LT_{2,1}^i & LT_{2,2}^i & LT_{2,3}^i & LT_{2,4}^i & LT_{2,5}^i & LT_{2,6}^i & LT_{2,7}^i & LT_{2,8}^i \\ LT_{3,1}^i & LT_{3,2}^i & LT_{3,3}^i & LT_{3,4}^i & LT_{3,5}^i & LT_{3,6}^i & LT_{3,7}^i & LT_{3,8}^i \\ H_{1,1}^i & H_{1,2}^i & H_{1,3}^i & H_{1,4}^i & H_{1,5}^i & H_{1,6}^i & H_{1,7}^i & H_{1,8}^i \\ H_{2,1}^i & H_{2,2}^i & H_{2,3}^i & H_{2,4}^i & H_{2,5}^i & H_{2,6}^i & H_{2,7}^i & H_{2,8}^i \\ H_{3,1}^i & H_{3,2}^i & H_{3,3}^i & H_{3,4}^i & H_{3,5}^i & H_{3,6}^i & H_{3,7}^i & H_{3,8}^i \\ AF_{1,1}^i & AF_{1,2}^i & AF_{1,3}^i & AF_{1,4}^i & AF_{1,5}^i & AF_{1,6}^i & AF_{1,7}^i & AF_{1,8}^i \\ AF_{2,1}^i & AF_{2,2}^i & AF_{2,3}^i & AF_{2,4}^i & AF_{2,5}^i & AF_{2,6}^i & AF_{2,7}^i & AF_{2,8}^i \\ AF_{3,1}^i & AF_{3,2}^i & AF_{3,3}^i & AF_{3,4}^i & AF_{3,5}^i & AF_{3,6}^i & AF_{3,7}^i & AF_{3,8}^i \\ Init_{1,1}^i & Init_{1,2}^i & Init_{2,3}^i & Init_{2,4}^i & Init_{2,5}^i & Init_{2,6}^i & Init_{2,7}^i & Init_{2,8}^i \\ Init_{2,1}^i & Init_{2,2}^i & Init_{2,3}^i & Init_{2,4}^i & Init_{2,5}^i & Init_{2,6}^i & Init_{2,7}^i & Init_{2,8}^i \end{bmatrix} \quad (1)$$

Where $L_{m,k}^i$ is the TLBO element, which determines the m^{th} bit of i^{th} a member of the population, the number of layers present in the Neural Network architecture, Opt_m^i determines the optimization algorithm to be employed during the training phase, with lr_m^i the learning rate. Similarly, $LT_{m,k}^i$ determines the type of hidden layer that is to

be used for k^{th} the layer. These vertical parameters are responsible for the creation of a hybrid deep neural network controlled by the CTO. However, the number of nodes in the hidden layers is determined by $H_{m,k}^{i}$ the $AF_{m,k}^{i}$activation function and $Init_{m,k}^{i}$the weight initializers.

The TLBO-based configuration of the population is given in Table 1. The CTO parameters-based configuration is depicted in Table 2. Table 2 describes the hidden layer configuration for a single layer. The proposed method has the ability to accommodate up to 8 hidden layers along with input and output layers.

Table 1. Teaching Learning Based Optimization configuration and its interpretation

S. No.	Bits	Parameters	Values	Parameter Value
1.	1-3	Number of Layers	000 001 010 011 100 101 110 111	1 2 3 4 5 6 7 8
2.	4-6	Optimizers	000 001 010 011 100 101 110 111	SGD Adam AdamW Adadelta RMSProp AdaGrad AdaMax Nadam
3.	7-8	Learning Rate	00 01 10 11	0.1 0.01 0.001 0.0001

Table 2. Class Topper Optimization configuration and its interpretation

S. No.	Bits	Parameters	Values	Parameter Value
1.	1-3	Type of Hidden Layer	000 001 010 011 100 101 110 111	DNN CNN RNN GRU LSTM BiLSTM Batch Normalization Dropout
2.	4-6	Number of Hidden Nodes ($H_{m,k}^{i}$)	000 to 111	$2^{H_{m,k}^{i}}$
3.	7-9	Activation Function	000 001 010 011 100 101 110 111	ReLU ELU CeLU Exponential Tanh Sigmoid Softplus Softmax
4.	10-11	Initializers	00 01 10 11	Random_Normal Random_Uniform Glorot_Normal Glorot_Uniform

These parameters, belonging to TLBO and CTO-controlled values, are discussed in the Methodology section under TLBO and CTO Parameters, respectively.

3.2. Teaching Learning Based Optimization

Rao et al proposed TLBO in [22]. The metaheuristic optimization algorithm is based on human cognitive ability of knowledge sharing. In this optimization algorithm, knowledge transfer from teacher to student is simulated on the solution space. The prospect solutions are initialized as given in equation 2:

$$pop = lb + randn \rightleftarrows ([0,1]) * (ub - lb) \quad (2)$$

Where pop is the population size, $N_p \times (D_1 \times D_2)$ where $N_p$is the size of the population, and $D_1 \times D_2$ is the dimension of each solution. Lower bound of the population is kept at all zeros, and upper bounds are kept at all ones. The randn function is utilized for random placement of solutions in the search space for the binary population.

The TLBO algorithm is a two-phase algorithm where, in the first phase, the knowledge is transferred from the teacher to the students. This phase is known as the teaching phase. In the teaching phase, the algorithms explore new solutions.

3.2.1. Teaching Phase

In this phase, the best student or solution is evaluated based on the fitness function. The solution with maximum accuracy was considered the best solution and was carried out to explore diverse regions of the search space. In our approach, to incorporate the binary search space, the modified equation for exploration of a new population in the teaching phase is given as:

$$X_{new} = X_{old} \oplus \{randn([0,1]) \bullet (X_{best} \oplus X_{mean})\} \quad (3)$$

Where$\oplus$denotes the XOR operation, which replicates the summing and difference operation of arithmetic in the Boolean problem. This is performed to neglect any carry or borrow bit in the new solution. X_{mean}Is the mean of the population in the current iteration, given by:

$$X_{mean} = \frac{1}{N_P}\left[\sum_{i=1}^{N_P} X_{old,i}\right] \quad (4)$$

In the above equation 4, the mean of the population is estimated to the lower bounds wherever applicable. This enables the authors to explore the search space uniformly, while keeping close to the best solutions. After new solutions have been found, the fitness of each new solution is evaluated. The greedy algorithm is applied to replace or preserve the old solution according to its fitness. If the fitness of the new solution exceeds that of older ones, the older solutions are replaced; otherwise, the new solution is discarded.

$$if(fit_{new}^{i} > fit_{old}^{i})$$
$$then(X_i \leftarrow X_{new}) and (fit_i \leftarrow fit_{new}) \quad (5)$$

3.2.2. Learning Phase

The next phase of the TLBO is the learning phase. In this phase, students are supposed to interact with each other and explore more knowledge. The students' interaction takes place in pairs. Each student is paired with another student in the given population. The partner selection process is carried out by a random integer pointing towards any member of the population in each iteration. The new solution explored by student X and the partner X_{ptr} is given as:

$$X_{new} = X \oplus \{randn([0,1]) \bullet (X \oplus X_{ptr})\} \quad (6)$$

The greedy selection is again used to replace the solution with the student having a better fitness value. These Teaching and Learning Phases are repeated for definite iterations. The best solution is found either by the Global best solution or the solution with the best fitness at the end of all iterations. In our model, the TLBO is used to find the most suitable depth of the neural network, along with the optimizer and learning rate, as discussed in Table 1.

3.3. Class Topper Optimization

Class Topper Optimization (CTO) is another optimization algorithm that mimics human learning behavior. The CTO was proposed by Das et al in [23], in which the authors emulated the learning and competitive approach seen in schools. The school has classes, and each class is divided into multiple sections. The algorithm is divided into a class level and into various sections. Section-level and student-level behaviors are replicated in search of an optimized solution. In the algorithm, the fitness of each student is evaluated during the Examination. Hence, each iteration is known as an examination. The initialization of the student is the same as the initialization performed by the TLBO algorithm in equation 1. The TLBO population consists of parameters for each hidden layer individually. Hence, the dimensions of the CTO population are different from those of the TLBO population in the proposed algorithm. The population is initialized with both TLBO and CTO populations combined as given in equation 2.

In the Section level of the CTO, each student attempts the Examination and their fitness, known as The Performance Index (PI), is evaluated as per the objective function. The student with the best PI is selected as the Section Topper (ST). Among the section toppers, the ST with the best PI, in turn, becomes the Class Topper (CT). In the proposed model, a single class with two sections has been considered for optimization. After every Examination, the students get a chance to enhance their grades. This step involves interaction of random students with the ST, and the new student solution is given as:

$$I_S^{E+1} = I_S^E \oplus de2bi(ST_{PI} \oplus S_{PI}) \quad (7)$$

Where I_S^E is the increment in student intellect while interacting with the ST? ST_{PI}, S_{PI} are the PIs belonging to the Student S and Section Topper ST? de2bi is used to convert the integer solution to binary. Random student or, in our case, a hidden layer is selected at random, and an update step is performed using equation 8 given as:

$$S_E = I_S^{E+1} \oplus S_E \quad (8)$$

In the next phase, the ST interacts with the CT, and the goal is to further enhance their PI and become the CT. This competition between ST and CT, considering that CT can only interact with ST but not with other Students, is given as:

$$I_{ST}^{E+1} = I_{ST}^E \oplus \rightleftarrows \rightleftarrows (de2bi(CT_{PI} \oplus ST_{PI})) \quad (9)$$

And the ST update takes place as:

$$ST_{E+1} = I_{ST}^{E+1} \oplus ST_E \quad (10)$$

Again, the greedy algorithm is employed to eliminate poor solutions. The step is emulated by comparing the PI of each section, and a new ST is selected. At the class level, all new STs are again compared to find out the new CT. The CT or multiple Class Toppers at the end of the maximum iterations provide the optimal solutions.

3.4. Objective Function

The objective of the proposed optimized hybrid Neural Network is to maximize the accuracy of the classifier implemented using TLBO and CTO series. The fitness function involving the accuracy of the classifier is given as:

$$fitness = max \frac{TP \rightleftarrows + TN}{TP+FP+FN+TN} \quad (11)$$

Where TP is the true positive, positive class instances are predicted as positive; similarly, TN correctly predicts negative class instances. On the other hand, FN and FP are falsely predicted positive and negative instances, respectively.

3.5. Teaching Learning Based Optimizer Series

The TLBO has been employed to optimize the external parametric values of the hybrid neural network structure. The following parameters, as depicted in Table 1, have been explained:

3.5.1. Number of Layers

The proposed algorithm has the ability to incorporate from 1 to 8 hidden layers. The first three bits of the TLBO population determine the number of layers used in the deep NN structure. The three bits can vary from 000, selecting a single hidden layer, to 111, selecting eight hidden layers, as shown in Table 1.

3.5.2. Optimizers

The proposed algorithm has given the users leverage to employ eight different optimizers in the hybrid neural network. Optimizers have application in the back-propagation of neural networks. Back-propagation is used to minimize the loss in the prediction of a class. In the training phase, the difference between predicted output and actual output is termed as loss. Different optimizers are employed for weight update as given:

SGD

The Stochastic Gradient Descent is selected for the 000 value in the TLBO series. The SGD is a variant of the gradient descent algorithm. It is an iterative method to compute the gradient of the loss function. This method has proved to be the most suitable for convex optimization problems. However, it is not suitable in case of higher order time, such as non-polynomial problems. The weight update step is given by:

$$\varsigma_{t+1} = \varsigma_t - \eta * \nabla_\varsigma(L) \quad (12)$$

Where ς are the weight, η the learning rate, and the loss gradient?

AdaGrad

It is an adaptive learning method for back-propagation and weight optimization in the NN architecture. As the name suggests, AdaGrad uses the gradient evaluated in the previous epochs. Suppose/is the diagonal matrix where the principal diagonal is the magnitude of the matrices formed by the gradient obtained in previous epochs. Then the weight update is given as:

$$\varsigma_{t+1} = \varsigma_t - \frac{\eta}{\sqrt{G_t+\varepsilon}} \cdot g_t \quad (13)$$

Where ε an arbitrarily small value is used to prevent a divide by zero error, and g_t is the gradient obtained in the present t^{th}epoch.

AdaDelta

The weight for AdaDelta uses the root mean square (RMS) value of the loss gradients given as:

$$\varsigma_{t+1} = \varsigma_t + \Delta\varsigma$$
$$where$$
$$\Delta\varsigma_t = \frac{-RMS(\Delta\varsigma_{t-1})}{RMS(g_t)} \cdot g_t \quad (14)$$

RMSProp

The weight update employs the total power of the gradient. The variance of any random number is defined as the second moment around the center.

$$\Delta\varsigma = -\frac{\eta}{\sqrt{E[g^2]+\varepsilon}} \cdot g_t \quad (15)$$

Adam

The adaptive moment estimation optimizer is the most commonly used optimizer nowadays. The Adam optimizer is computationally efficient for large data. The optimizer uses alpha and beta parameters with the mean and standard deviation of the loss gradient to update weights.

$$\varsigma_{t+1} = \varsigma_t - \frac{\eta}{\sqrt{\sigma_t + \varepsilon}} \cdot \mu_t \quad (16)$$

Where σ_t is the standard deviation, and μ_t Is the mean of the loss gradient. The mean and standard deviations are updated after every epoch, given as:

$$\mu_t = \alpha \cdot \mu_{t-1} + (1-\alpha) \cdot g_t$$
$$and$$
$$\sigma_t = \beta \cdot \sigma_{t-1} + (1-\beta) \cdot g_t^2 \quad (17)$$

AdamW

Weighted Adam optimizer incorporates negative effects of μ_t and g_t λ is used as an arbitrary positive real number to negotiate weight decay.

$$\varsigma_{t+1} = \varsigma_t - \eta \cdot \varsigma_{t-1} - \frac{\eta}{\sqrt{\sigma_t + \varepsilon}} \cdot \mu_t \quad (18)$$

AdaMax Optimizer

The AdaMax optimizer replaces the L2 regularization used in the Adam optimizer. The L2 norm is replaced by the L∞ norm in the AdaMax optimizer. The AdaMax optimizer uses the mean and standard deviation, the same as the Adam optimizer, and the weight update is given as:

$$\varsigma_{t+1} = \varsigma_t - \eta \cdot \frac{\hat{\mu}_t}{u(t)} \quad (19)$$

Where u(t) is given by:

$$u(t) = max(\beta \cdot u(t-1), |g_t|) \quad (20)$$

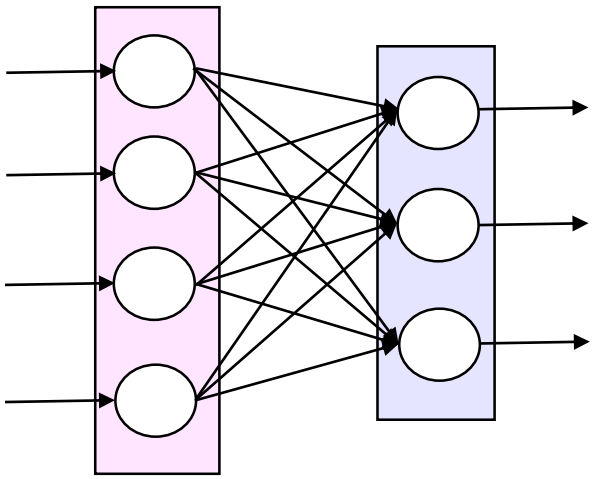

Fig. 1 Pictorial Representation of the Fully Connected Network

Nadam Optimizer

The Nadam optimizer uses the Nesterov update system along with the Adam optimizer, given as:

$$\varsigma_{t+1} = \varsigma_t - \eta \cdot \frac{\beta \cdot \hat{\mu}_t + (1-\beta) \cdot g_t}{\sqrt{\sigma_t + \varepsilon}} \quad (21)$$

3.5.3. Learning Rate

The learning rate, as seen with the optimizer, decides the rate at which the gradient function influences the weights. The proposed algorithm uses the TLBO series to determine the learning rate. The last bits of the TLBO series are converted to decimal numbers. The decimal equivalent, let us say, *lr* gives the learning rate as:

$$\eta = \frac{1}{10^{lr}} \quad (22)$$

3.6. Class Topper Optimization Series

The class topper optimization series determines the internal parameters of each layer. The CTO Series is initialized for the maximum possible layers, and the layers that undergo training are determined by the TLBO series. The internal parameters, such as the type of layer, number of hidden nodes for each layer, activation function and initializers, are determined using a series. The following possible outcomes are listed below:

3.6.1. Type of Hidden Layer

The CTO series determines the type of hidden layers for training and testing for malicious packets in the proposed algorithms. Different types of layers contribute to different ways of extracting features from the data. In our proposed method, feature extraction is performed using a metaheuristic method and normalization and feature deletion can also be performed.

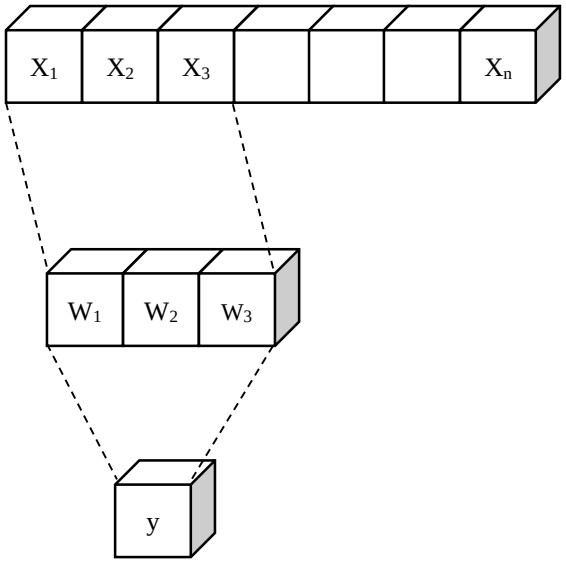

Fig. 2 Pictorial representation of CNN layer with x input, w moving window depicting convolution to give y output.

DNN

This is the basic layer of any multi-layer perceptron. The DNN consists of Fully Connected Network Layers (FCN). In this type of layer, every neuron of the preceding layer is connected to the DNN layer. The output y of the DNN layer with weights ω and bias b is given by:

$$y = \Im(\omega \cdot x + b) \quad (23)$$

Where x is the input, and $\mathfrak{I}$ what is the activation function? The FCN is generally used in the output layer, as it ensures that all the information of the extracted features is passed to the decision device. According to the universal approximation theorem, FCN can approximate any continuous function; therefore, FCN becomes a reliable layer for feature extraction. The pictorial representation of the FCN is given in Figure 1.

CNN

Basically, invented for image classification and segmentation, Convolutional Neural Network (CNN) has found many applications in signal processing, Natural Language Processing (NLP), and several other fields. As the name suggests, the CNN uses convolution of a moving window having a kernel size $(1 \times \kappa)$ for signal processing with a stride of length $\mathcal{S}$. The convolved output is used for feature extraction in deep CNN architectures. The CNN is capable of dealing with high-dimensional data. In this paper, the strides $\mathcal{S}$ are set to 1 to make the output compatible with other layers.

$$y = \Im(\omega * x + b) \quad (24)$$

Where * symbol represents that the weight window is convolved with the input sequence instead of the dot product, as shown in equation 23. Figure 2 gives the pictorial representation of the CNN layer.

RNN

The FCN and CNN extract features only in the forward direction; hence, they are called feed-forward networks. These layers lack feedback capabilities. To incorporate such features and succeeding layers that have an impact on the preceding layer, feedback-type layers have gained popularity. The feedback feature of RNN enables it to create a loop through different layers. This is achieved using temporary units. RNNs, although capable of memory retention, suffer from the problem of vanishing gradients. The output of the RNN layer is given by:

$$y = \Im([\omega_{hx} \cdot x_p + \omega_{hh} \cdot h_{p-1}] + b) \quad (25)$$

The self-feedback loop of the RNN is represented in Figure 3.

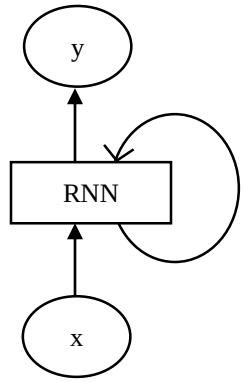

Fig. 3 Feedback Representation of the RNN layer

LSTM

Like GRU, Long-Short-Term Memory (LSTM) was invented to mitigate the vanishing gradient problem. The LSTM layer uses a memory cell and three gates instead of two, in the case of GRU. The LSTM layer requires a Reshape layer when used in hybrid models. Explicit reshape layers were added to make LSTM compatible with the hybrid architecture.

Batch Normalization

The Batch Normalization uses z normalization of the extracted features. The Batch normalisation restores the feature bounds and, as a result, the learning and discrimination of features is accelerated. Feature normalization is given by:

$$x_{norm} = \frac{x-\mu}{\sigma} \quad (26)$$

Where μ is the average value, and σ^2 is the variance of the feature vector x. The scaling and shifting parameters α,β respectively, are employed for output measurement. The scaled and shifted vector is given by:

$$y = \alpha \cdot x_{norm} + \beta \quad (27)$$

Dropout

For feature selection, a dropout layer is used as an optional layer. In our model, it was coded to prevent consecutive dropout layers. 25% of the least important features were dropped using this layer. This layer is added to prevent the neural network from overfitting.

3.6.2. Number of Hidden Nodes

Each layer has a different number of nodes, filters or units depending on the type of layers. This is controlled by bits 4 to 6 of the CTO series for every hidden layer. Generally, the number of nodes is an integer power of 2. Hence, in this paper, the three bits are converted to an integer equivalent, and their power of 2 creates the number of nodes. The smallest value, 0's for all three bits, represents a single

node hidden layer, whereas the maximum nodes can be 128 with three bits as 1's.

3.6.3. Activation Function

Activation functions are used in Neural networks to apply non-linearity to the layer output. This is performed to ensure projection of non-linear practical data in a non-linear suitable feature space. Without activation functions, the deep learning algorithms would be mere linear functions of inputs that restrict the use of deep learning only to ideal scenarios.

ReLU

Rectified Linear Unit is a highly utilized activation function in deep learning algorithms. The ReLU activation allows the positive and negative features to be nullified, restricting the negative features from reaching the output layers. Transfer learning and pre-trained model, such as AlexNet, have employed ReLU as their activation function. The transfer function of ReLU is given as:

$$Re\,L\,U = \begin{cases} x, X \geq 0 \\ 0, otherwise \end{cases} \quad (28)$$

ELU

It stands for exponential linear unit. It has an advantage over ReLU in that the ELU uses negative and positive features. ELU is even preferred over softmax and sigmoid, as these activation functions tend to saturate at a boundary. Unlike sigmoid, ELU can extend up to infinity.

$$ELU = \begin{cases} x, X \geq 0 \\ \alpha \cdot (e^{x} - 1), otherwise \end{cases} \quad (29)$$

Exponential

As the name suggests, the output of the activation function is the exponential power of the input for all real values.

Tanh

This activation function is the default and the only allowed activation function for feedback-type layers. The function acts as a linear function for smaller values of inputs, but quickly saturates to ±1 for larger values. The function is given as:

$$tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (30)$$

Sigmoid

This activation function is used for the binarization of features. Generally, this activation is utilized at the output layer for binary classification. This activation simulates the Fermi probability of any occupied state. The transfer function confined between 0 and 1 is given as:

$$sigmoid = \frac{1}{1 + e^{-x}} \quad (31)$$

Softplus

Similar to sigmoid, softplus employs the exponential power of inputs. The softplus uses a logarithmic function, making it suitable for large feature values. The softplus is given as:

$$softplus = log(1 + e^{x}) \quad (32)$$

In the above equation 32, a one is added to the exponential input to avoid a log of zero error.

CeLU

It stands for a continuously differentiable linear unit. As the name suggests, the transfer function is continuous and differentiable at each point. The CeLU was proposed by Barron in [24]. The transfer function is given as:

$$ELU = \begin{cases} x, X \geq 0 \\ \alpha \cdot (e^{x/\alpha} - 1), otherwise \end{cases} \quad (33)$$

Softmax

It is used in the output layer of multiclass classification problems. The softmax converts the features into a probabilistic feature space. The probabilities of the features are compressed between 0 and 1.

Figure 4 depicts various activation functions for a range of inputs.

3.6.4. Initializers

The TLBO series uniquely provides each layer with its own initializer. Random as well as Glorot initializer are available in the CTO series. The Random initializer randomly initialize weights as either uniformly or normally distributed random numbers. At the same time, the Glorot initializer takes the fan-in and fan-out of the layer into consideration while initializing the weights.

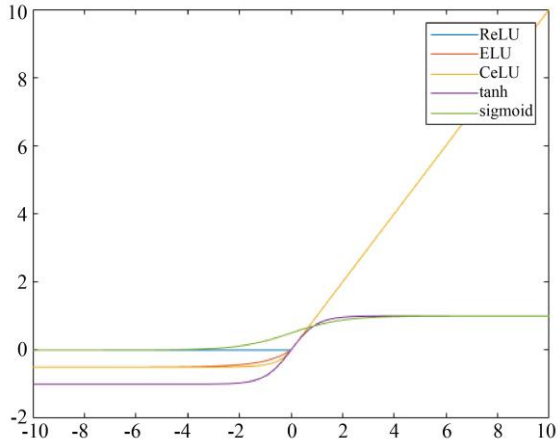

Fig. 4 The transfer function of different activation functions for a range of inputs from -10 to 10

4. Experimental Results and Discussions

The flowchart of the experiment is depicted in Figure 9. The experiment is carried out on a Dell Precision 5820 workstation with Windows 10 and Python 3.8. Tensorflow with the keras library is used to train the deep learning models. The CPU configuration of the system is 2TB HDD, 16GB RAM DDR5 and 4GB GPU of Ryzen.

The experiment starts with reading the data. The NSL-KDD dataset is employed in this experiment to train and test the explored hybrid neural networks. The dataset is a refined version of the KDDCup99 dataset. The KDD99 dataset was cleaned by removing redundant data, and duplicates were also removed by Tavalaee [25].

The dataset consists of 41 features and 125,973 instances. The dataset was made by DARPA in 1999. The dataset contains 67,343 normal instances, and the rest are attack instances. These attack instances were recorded for 24 different attack strategies, such as ping of death, smurf, saint, warezmaster, etc.

These 24 attacks are marked as class 1, and normal instances are marked as class 0 for binary classification. In comparison, these attacks were marked under their umbrella attack strategies. Denial of Service (DoS), where an attacker disrupts the network resources, was labelled as 1. Probe, which is a passive attack strategy, was labelled 2. User to Root (U2R), in which the intruder gains super user access and Remote to Local (R2L), where an outside masquerades as the network, were labelled 4 and 5, respectively, for multiclass classification. This step is known as class labelling.

Out of 41 features in the dataset, three of them are alphanumeric in nature. These features are protocol type, which has information about the communication protocol. Service and flag features are other alphanumeric features present in the data. These features are converted into a numeric value using a label encoder.

In the optimization process, the first step involves the initialization of the population. Each solution has a size of 12x8. The given size is the combined size of the TLBO and CTO. The TLBO series has a single row with eight elements, whereas CTO consist of 11 elements belonging to 8 possible layers. This makes the CTO series 11x8 in size. The population of students is taken to be 10. Hence, 10x12x8 becomes the initial population size.

In the next step, a random decision is made on the model that either structural or hidden layer parameters are modified to explore a new solution. A random variable is generated; if this variable is less than 0.5, the TLBO algorithm is implemented; otherwise, the CTO algorithm is implemented. This gives equal opportunities to both TLBO and CTO.

If the TLBO is implemented, the external parameters are modified by the teaching phase and learning phases as given in equations 3 to 5 for the teaching phase and equation 6 in the learning phase. As the population is binary, these equations are modified by employing AND and XOR Boolean operations.

If the CTO is selected, a random hidden one is selected from the two sections and modified using interaction with the class topper as well as the section topper.

The modified population is then interpreted as a Hybrid neural network. The OHNN is then trained and tested on the NSL-KDD dataset. The OHNN with higher accuracy replaces the inferior solution in each iteration. The optimization was run for 100 iterations. At the end of the iteration, the best architecture is obtained.

4.1. Binary Classification of NSL-KDD Dataset

The experiment is performed on the NSL-KDD dataset with normal as class 0 and all other classes as 1. The Binary classifier is a six hidden structure. The OHNN consists of an input layer with 41 features, followed by a CNN layer with 64 nodes, ReLU activation and Random Uniform Initializer. This layer is followed by an RNN layer with 16 units, tanh activation and Glorot Normal initializer.

The RNN layer is followed by a DNN layer with 16 nodes, a ReLU activation function and a Random Normal initializer. After that, an LSTM layer with 32 units, tanh, and a Random Normal initializer was modelled by OHNN. These layers were followed by DNN and RNN layers with CeLU and tanh activation functions. Both layers were initialized using random uniform initializers.

The output layer is always a Dense layer with 2 nodes for binary classification. The Global Parameters neglected the last two layers of the CTO series by selecting a 6-layer deep architecture, making them non-critical or redundant layers. The Adadelta optimizer was used to update the weights of the OHNN with a 0.01 learning rate. The Binary OHNN achieved 94.33% accuracy. The figure depicts the accuracy of binary OHNN over 50 epochs.

4.2. Multiclass Classification of NSL-KDD Dataset

The multiclass classifier using OHNN was found to be a 7-layered structure. The structure has an input layer with 41 features of KDD.

The first hidden layer is an LSTM layer with 32 units, tanh activation and a random normal initializer. This layer was followed by two CNN layers with 32 and 64 nodes, respectively. Glorot uniform and random normal initializer were selected.

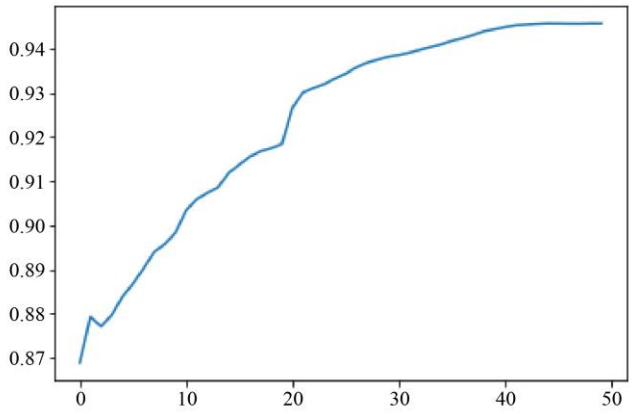

Fig. 5 Binary OHNN accuracy vs epochs

The next four layers are feedback-type layers. The four layers are LSTM, followed by RNN and two LSTM layers. The layers diverged with 16, 32 and 128 units, followed by 16 units. All feedback layers are hard-coded to have a tanh activation function. The first LSTM and RNN have a Random Uniform initializer, followed by Random Normal, an initialiser followed by another Random Uniform. Reshape layers were explicitly added to the network.

The output for the multiclass classifier has 5 nodes for 5 classes with softmax activation. The TLBO series selected the AdaMax optimizer with a 0.01 learning rate. The 7-layered architecture gave 99.37% accuracy, as shown in Figure 6, and the model plot is shown in Figure 8.

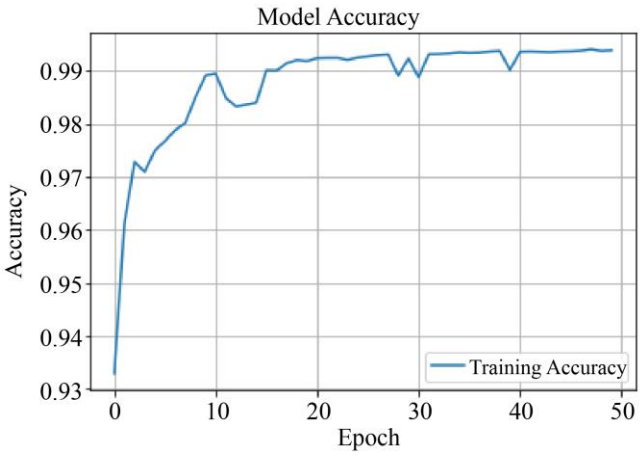

Fig. 6 Multiclass OHNN Accuracy vs Epochs Curve

4.3. Comparison with Existing Models

A comparison with existing research papers is given in Table 3. The proposed algorithm outperformed several machine learning and deep learning algorithms. The TLBO and CTO optimized the architecture. After 100 iterations, the binary classifier has 6 6-layer structure as shown in the

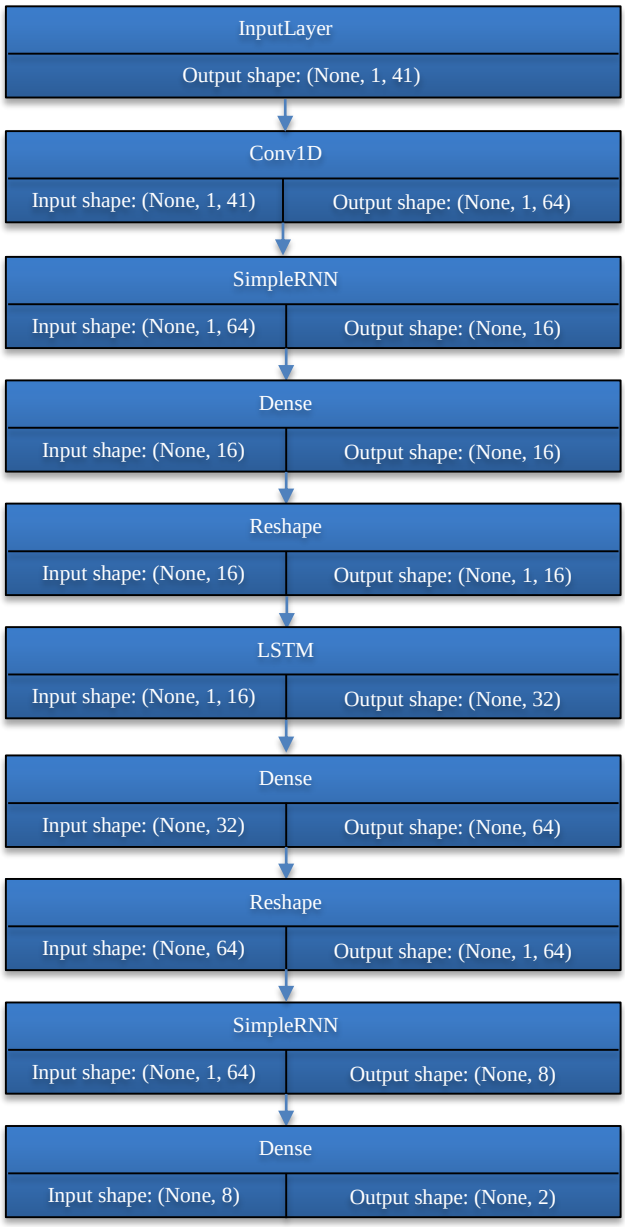

Fig. 7 OHNN structure for Binary Classification

Figure 7 The exploration and hyper-parameter tuning gave an accuracy of 94.33%. The CNN layer, in combination with feedback networks, RNN and LSTM, has led to the extraction of moving windowed features utilizing the feedback effect of the succeeding recurrent layers.

Thus, improving the accuracy of the model. These features are then passed to the dense layer for the extraction of linear as well as non-linear feature approximation. Due to the novel twin optimization strategy, the proposed architecture outperformed several existing methodologies, as shown in Table 3.

Table 3. Comparison with SOTA Algorithms

S. NO.	Author	Technique	Accuracy
1.	VinayKumar	Deep	75.20
2.	[26]	Learning	93.20
3.	Almeseidin [27]	J48	90.30
4.	Ingre [28]	DT	98
5.	Jin [29]	Rule Based	**94.33**
6..	**OHNN Binary OHNN Multi**	**Presented Bin Presented multi**	**99.37**

For multiclass classification of intrusions, not only the malicious packets but also the type of attack are determined. The optimized TLBO-CTO model gave a predominantly recurrent type architecture for the multiclass classification.

The LSTM and RNN layers are capable of extracting long-term features, but they lack a local understanding of features, which were indeed extracted by the CNN layers, creating an interface between two recurrent sections.

The CNN and recurrent layer complement each other, making an optimal architecture for detection purposes. Table 3 depicts that the optimized architecture outperformed the existing techniques by a large margin.

5. Conclusion

In this paper, the TLBO and CTO-based optimized hybrid neural network has been proposed. In this algorithm, the human requirement for designing a deep learning algorithm is considered. The TLBO was appointed for outer parameters such as the number of layers, optimizers and learning rate.

The CTO was appointed as an internal parameter selector. These include the type of layer, number of hidden nodes or units, activation function and weight initializers. The proposed optimization algorithm was iterated 100 times with 10 student population size.

The binary classifier was a 6-layer structure with 94.33% accuracy, whereas the multiclass 7-layer structure gave 99.37% accuracy.

In future, other metaheuristic algorithms such as bio-inspired grey wolf optimization, ant colony optimization, etc., can be employed for exploration of new deep architectures. In addition, some of the pre-trained models, such as ResNet, VGG16, VGG19, etc., can also be taken into consideration.

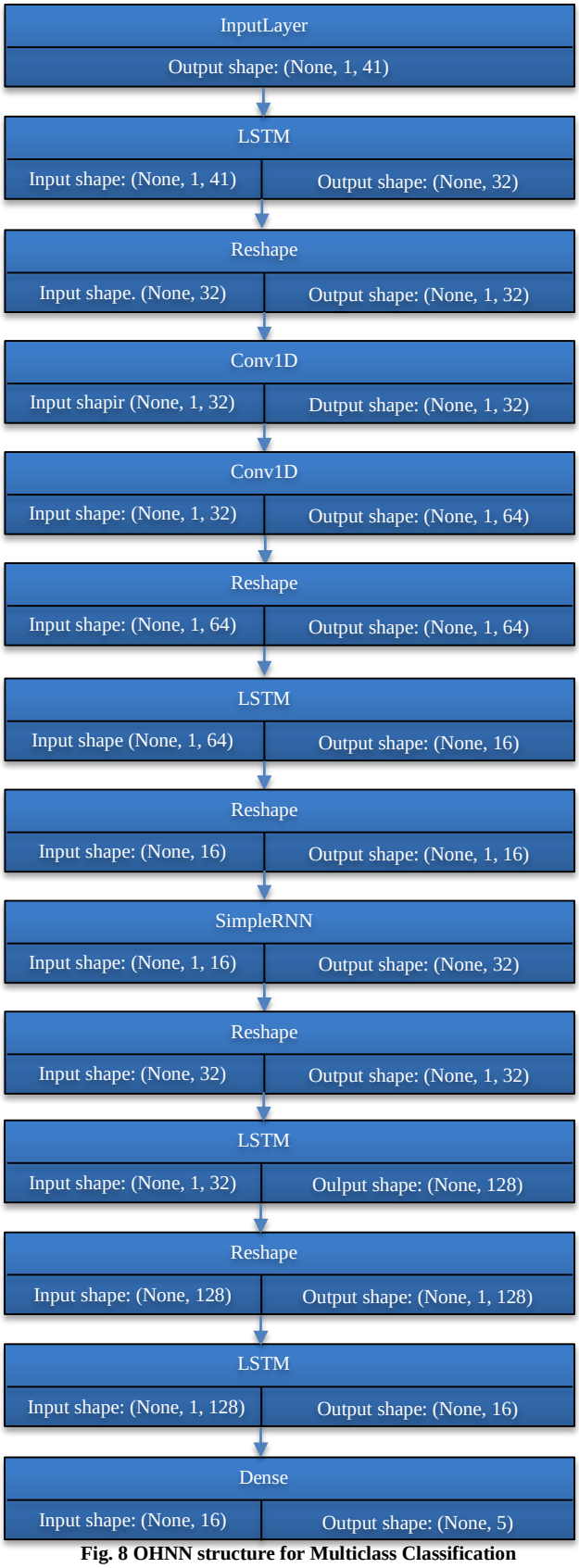

Fig. 8 OHNN structure for Multiclass Classification

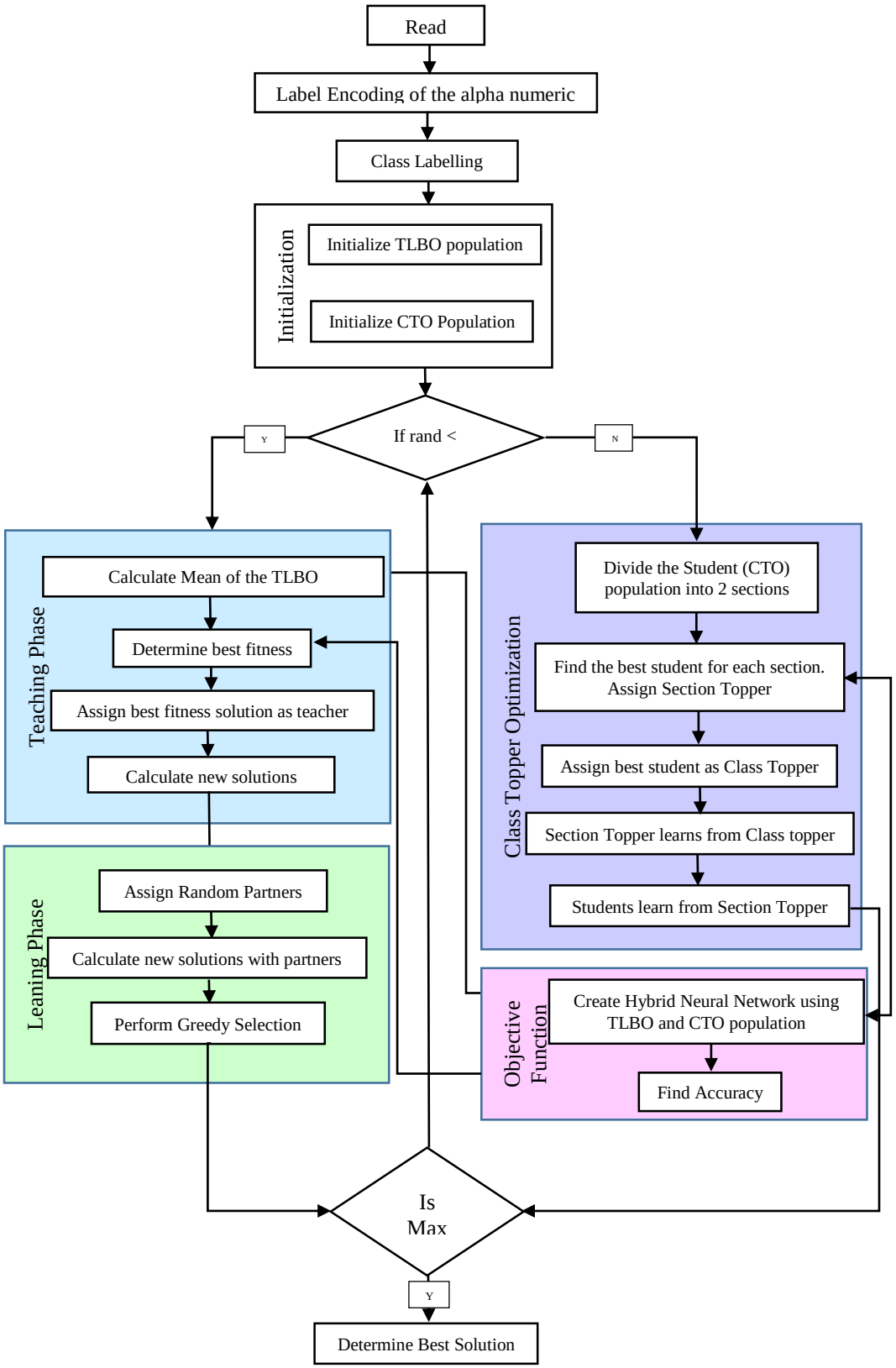

Fig. 9 Flowchart of the proposed algorithm

References

[1] Salman Muneer et al., "A Critical Review of Artificial Intelligence Based Approaches in Intrusion Detection: A Comprehensive Analysis," *Journal of Engineering*, vol. 2024, no. 1, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Abdul Qaddos et al., "A Novel Intrusion Detection Framework for Optimizing IoT Security," *Scientific Reports*, vol. 14, pp. 1-22, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[3] Albara Awajan, "A Novel Deep Learning-Based Intrusion Detection System for IoT Networks," *Computers*, vol. 12, no. 2, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] Emad Ul Haq Qazi, Muhammad Hamza Faheem, and Tanveer Zia, "HDLNIDS : Hybrid Deep-Learning-Based Network Intrusion Detection System," *Applied Sciences*, vol. 13, no. 8, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Md. Alamgir Hossain, and Md. Saiful Islam, "Ensuring Network Security with a Robust Intrusion Detection System using Ensemble-Based Machine Learning," *Array*, vol. 19, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Mansi Bhavsar et al., "Anomaly-based Intrusion Detection System for IoT Application," *Discover Internet of Things*, vol. 3, pp. 1-23, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Zihan Wu et al., "RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System," *IEEE Access*, vol. 10, pp. 64375-64387, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Sydney Mambwe Kasongo, "A Deep Learning Technique for Intrusion Detection System Using a Recurrent Neural Networks based Framework," *Computer Communications*, vol. 199, pp. 113-125, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Mohamed Selim Korium et al., "Intrusion Detection System for Cyberattacks in the Internet of Vehicles Environment," *Ad Hoc Networks*, vol. 153, pp. 1-16, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Dhiaa Musleh et al., "Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT," *Journal of Sensor and Actuator Networks*, vol. 12, no. 2, pp. 1-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Hakan Can Altunay, and Zafer Albayrak, "A Hybrid CNN+LSTM based Intrusion Detection System for Industrial IoT Networks," *Engineering Science and Technology, an International Journal*, vol. 38, pp. 1-13, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] G. Logeswari, S. Bose, and T. Anitha, "An Intrusion Detection System for SDN Using Machine Learning," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 867-880, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ajay Kaushik, and Hamed Al-Raweshidy, "A Novel Intrusion Detection System for Internet of Things Devices and Data," *Wireless Networks*, vol. 30, pp. 258-294, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Salam Fraihat et al., "Intrusion Detection System for Large-Scale IoT NetFlow Networks Using Machine Learning with Modified Arithmetic Optimization Algorithm," *Internet of Things*, vol. 22, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Abdullah Alzaqebah et al., "A Modified Grey Wolf Optimization Algorithm for an Intrusion Detection System," *Mathematics*, vol. 10, no. 6, pp. 1-16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Dusmurod Kilichev, and Wooseong Kim, "Hyper-Parameter Optimization for 1D-CNN-Based Network Intrusion Detection Using GA and PSO," *Mathematics*, vol. 11, no. 17, pp. 1-31, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Yesi Novaria Kunang et al., "Attack Classification of an Intrusion Detection System using Deep Learning and Hyper-Parameter Optimization," *Journal of Information Security and Applications*, vol. 58, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Muhammad Imran et al., "Intrusion Detection in Networks using Cuckoo Search Optimization," *Soft Computing*, vol. 26, pp. 10651-10663, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Shahid Latif et al., "DTL-IDS: An Optimized Intrusion Detection Framework using Deep Transfer Learning and Genetic Algorithm," *Journal of Network and Computer Applications*, vol. 221, pp. 1-10, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Subham Kumar Gupta, Meenakshi Tripathi, and Jyoti Grover, "Hybrid Optimization and Deep Learning Based Intrusion Detection System," *Computers and Electrical Engineering*, vol. 100, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Doaa Sami Khafaga et al., "Voting Classifier and Metaheuristic Optimization for Network Intrusion Detection," *Computers, Materials & Continua*, vol. 74, no. 2, pp. 3183-3198, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[22] R.V. Rao, V.J. Savsani, and D.P. Vakharia, "Teaching–Learning-Based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303-315, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[23] Pranesh Das, Dushmanta Kumar Das, and Shouvik Dey, "A New Class Topper Optimization Algorithm with an Application to Data Clustering," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 4, pp. 948-959, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jonathan T. Barron, "Continuously Differentiable Exponential Linear Units," *arXiv Preprint*, pp. 1-2, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[25] Mahbod Tavallaee et al., "A Detailed Analysis of the KDD CUP 99 Data Set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp. 1-6, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[26] R. Vinayakumar et al., "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525-41550, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[27] Mohammad Almseidin et al., "Evaluation of Machine Learning Algorithms for Intrusion Detection System," *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, Subotica, Serbia, pp. 277-282, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[28] Bhupendra Ingre, Anamika Yadav, and Atul Kumar Soni, "Decision Tree Based Intrusion Detection System for NSL-KDD Dataset," *Information and Communication Technology for Intelligent Systems*, vol. 2, pp. 207-218, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[29] Dongzi Jin et al., "SwiftIDS: Real-Time Intrusion Detection System based on LightGBM and Parallel Intrusion Detection Mechanism," *Computers & Security*, vol. 97, 2020. [CrossRef] [Google Scholar] [Publisher Link]