IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 May 2023, accepted 24 May 2023, date of publication 1 June 2023, date of current version 6 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3281893

RESEARCH ARTICLE

A Fully Streaming Big Data Framework for Cyber Security Based on Optimized Deep Learning Algorithm

NOHA HUSSEN[1], SALLY M. ELGHAMRAWY[2], (Senior Member, IEEE), MOFREH SALEM[1], AND ALI I. EL-DESOUKY[1]

[1]Computers Engineering and Systems Department, Mansoura University, Mansoura 35516, Egypt
[2]Computer Engineering Department, Misr Higher Institute for Engineering and Technology, Mansoura 31527, Egypt

Corresponding author: Sally M. Elghamrawy (sally_elghamrawy@ieee.org)

ABSTRACT Real-time deep learning faces the challenge of balancing accuracy and time, especially in cybersecurity where intrusion detection is crucial. Traditional deep learning techniques have been insufficient in identifying network anomalies and intrusions. To address this, a Fully Streaming Big Data Framework based on optimized Deep Learning for cybersecurity (FSBDL) was proposed. The framework leverages two parallel optimization algorithms, Adam and RMSprop, labeled Hyper-parallel optimization (HPO) techniques to enhance efficiency and stability. The optimized CNN in the proposed framework achieves high accuracy in real-time intrusion detection without compromising reliability. The CNN is customized to address overfitting issues in recurrent connections by reducing the number of training parameters, using customized activation functions and regularization techniques. The CNN is trained in parallel using Adam and RMSprop optimization algorithms, resulting in significant accuracy improvements that surpass traditional methods and current state-of-the-art approaches. The HPO is a crucial component of the proposed framework, as it enables the system to detect intrusions in real-time, ensuring prompt response to potential cyber threats. The six-layer FSBDL framework includes data pre-processing, feature selection, hyper-parallelism, a customized CNN, a GUI layer for interpretation, and a detection-evaluation layer. The optimized CNN was designed to detect intrusions in real-time without compromising accuracy or reliability. The proposed algorithms were evaluated using various performance metrics, showing that the accuracy of the framework surpasses 99.9%, indicating its superiority over other intrusion detection models. This novel intrusion detection model offers promising prospects for cybersecurity, and its effectiveness and accuracy have been demonstrated through experimentation.

INDEX TERMS Cyber security, streaming data, intrusion detection, deep learning, conventional neural network (CNN), optimization.

I. INTRODUCTION

With the proliferation of streaming data and advanced technical processing, the magnitude of network and traffic complexity is on the rise. As a result, navigating and managing these systems requires increasingly sophisticated skills and strategies. Streaming data generated by computer networks changes over time and arrives infinitely at high speeds. To this end, intelligent intrusion detection systems must rapidly monitor and analyze this growing data stream in real-time [1].

Recent research has classified real-time IDS into three categories based on the method's objective. The primary objective of the initial selection technique is to reduce the duration required to arrive at a verdict. The second focuses on pattern matching for dedicated hardware to speed up intruder detection, which often results in better real-time performance. The third type is based on a combination of

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

different big data techniques applied by machine learning algorithms [2].

When considering IDS as a binary problem, the conduct of the access network is classified as either non-malicious or malicious. There are several kinds of attacks, and network traffic is one of the lines that an attacker might penetrate. The second perspective considers the network traffic segmented into four distinct classes: Normal DoS (Denial of Service), R2L (Root to Local), U2R (User to Root and Probe) [3], [4].

Many traditional machine learning techniques are used to identify network anomalies, but due to their flat architecture and inability to handle larger datasets, they are not suitable for big data environments and cannot detect intruders in real time [5]. The utilization of DL has become increasingly widespread across a wide range of applications [6], [7], [8], [9], [10]. The effectiveness and accuracy of DL frameworks can be greatly improved by selecting an appropriate training algorithm and fine-tuning various factors and percentage of false detections, and functions used for training [11]. Furthermore, the performance of IDS can be improved by choosing the best classification technique and feature extraction method [12], [13].

Deep Learning (DL) methodologies such as Long Short-Term Memory (LSTM) and Support Vector Machines (SVM) have also been used for IDS [14], particularly multi-layered Convolutional Neural Networks (CNNs), have gained popularity in the field of Intrusion Detection Systems (IDS).

The implementation of multi-layered CNN has been used in the use of DL methodologies for IDS [15]. The proposed model has limitations, such as relying heavily on labeled data and being optimized for use in massive networks, which may reduce its effectiveness in smaller networks or different use cases. Despite this, CNNs in the model can classify and select features in traffic data and even learn better features than traditional feature selection algorithms. Additionally, the use of similar convolutional kernels in CNNs reduces computational complexity and the number of parameters required for training, making it easier to identify types of traffic data attacks [16]. However, there are challenges in applying an IDS in real-time [17]. including the need to anticipate changes in attacker behavior patterns that may bypass the IDS, the importance of data features in training the model, and the need for performance evaluation to achieve high detection in real-time.

Another crucial aspect that can influence the effectiveness of a DL model is the choice of an optimizer. The optimizer is crucial in adjusting the model's weights and biases during training influencing the model's accuracy and efficiency.

As deep learning research has progressed, the development of efficient optimization algorithms has become increasingly important. Among the many optimization algorithms that have been proposed, Adam and RMSprop have emerged as popular choices for training deep neural networks.

Adam, which stands for Adaptive Moment Estimation [18], is an optimization algorithm that uses gradient-based first-order optimization and tracks the mean of the parameters to determine a running estimate of the second raw moment of the gradient; The adaptive learning rate adjusts the learning rate. On the other hand, viewed from another perspective, RMSprop is a gradient-based second-order optimization algorithm that uses the average of the squared gradients over time to determine an appropriate learning rate for each parameter [19], [20].

By combining these two optimization algorithms, the model can benefit from the fast convergence and adaptive learning rate of Adam, as well as the stability and precision of RMSprop. This can lead to better performance and accuracy in tasks related to cyber security, such as detecting and classifying cyber threats or predicting vulnerabilities in systems. The combination of Adam and RMSprop can also help to prevent overfitting and improve generalization, which is critical for effective cyber security applications.

II. MOTIVATIONS AND CONTRIBUTIONS OF THE STUDY

The purpose of this research is to improve the efficiency of the Fully Streaming Big Data Framework for Cyber Security (FSBDL) by integrating two parallel optimization algorithms, Adam and RMSprop.

The motivation behind using these two parallel optimization techniques is to enhance the optimization process's convergence and efficiency. These techniques use different approaches to fine-tune the learning rate of the model, which can help mitigate the issue of oscillation or divergence during training. Using two parallel optimization techniques can also provide a stronger optimization process, as the different approaches may be more effective in different situations.

The objective ultimately is to identify the most suitable set of parameters for the model with maximum speed and precision. The FSBDL model is based on six layers: data processing, feature selection, extraction, Hyper parallel optimization. data flow deep learning, GUI design, and detection-evaluation. In this study, two parallel optimization techniques Adam and RMSprop were integrated to optimize the big data used in the FSBDL system and address the issue of redundant data in the NSL-KDD dataset through a hybrid solution [21]. We designed a GUI for real-time IDS. Our ultimate goal is to enhance the FSBDL framework's performance concerning accuracy, precision, and sensitivity and specificity compared to other schemes.

The main contributions of the paper are as follows:

- Propose a Fully Streaming Big Data Framework based on optimized DL for cyber security (FSBDL) that enhance the accuracy of real-time intrusion detection systems.
- Proposed a parallel optimization technique that utilizes both Adam and RMSprop algorithms to enhance the model's accuracy, stability, and generalization ability.
- Introduce techniques for optimizing CNN algorithms to handle imbalanced NSL-KDD data.

- Evaluate the performance of the deep learning algorithms by assessing their accuracy, precision, specificity, and sensitivity rankings.
- Implement a GUI to aid interaction and timely detect network behavior.
- Compare the results of the proposed framework with existing algorithms.

The paper's roadmap is as follows: Section II discusses the justification and significance of the study. Section III provides a comprehensive examination of relevant literature, including a summary of CNNs and ID. Section IV introduces the proposed FSBDL and the developed framework. Section V analyses and discusses the results of the experiments. Section VI. presents the study's conclusion and suggestions for future work.

III. OVERVIEW AND RELATED WORK

Big data has increased the number of cyber intrusion attacks, as many studies have noted [16], [21], [22]. Attackers hide activities, making it hard to configure IDS, especially for zero-day attacks without prior knowledge [23].

Real-time intrusion detection systems are vital for preventing network attacks. Researchers identify three key issues: reducing detection time, enhancing accuracy with filter-based feature selection, and improving accuracy and reducing false alarms with deep learning algorithms [24], [25].

Feature selection reduces dimensionality and enhances computational efficiency of detection algorithms by identifying significant features using statistical measures such as correlation and mutual information. In [26], a novel feature selection method for network IDS is proposed that employs combinatorial optimization. The proposed approach aims to improve the accuracy of IDS by identifying the most relevant and meaningful features from large datasets.

The second category of real-time ID involves filter-based feature selection methods that evaluate the relevance of each feature independently of the classification algorithm. These methods, which include chi-square testing, information exchange, and correlation-based selection of features, enhance detection accuracy and are appropriate for large feature spaces. In a recent study [22], The author suggested a new fog-enabled IoT network attack detection system utilizing filter-based feature selection approaches to improve attack recognition accuracy by picking the most relevant features from a vast dataset.

The third category of real-time intrusion detection involves using deep learning algorithms to improve accuracy and reduce false alarms. These algorithms can learn relevant features from raw data and handle complex and nonlinear relationships between input features and target variables.

Various deep learning models such as CNNs, RNNs, and DBNs have been used for real-time detection in different domains [23]. A new intrusion detection approach using FHD-CNN was proposed in [27], highlighting the limitations of traditional systems and the need for carefully curated training data and constant updates to adapt to evolving threats. A method for assessing machine degradation was suggested in [28] using DCNN and transfer learning techniques. In [20], To increase accuracy, an optimized ANN-based technique was developed by investigating alternative hyperparameters, applying ensemble methods, and using larger and diverse datasets. Deep learning techniques have been presented to boost the accuracy of ID in a variety of domains. The paper proposed in [29] suggests using a combination of DNN and transfer learning to improve the performance of the intrusion detection system in real-time. In [30], the authors propose a new intrusion detection system that uses audio signals and convolutional neural networks for classification. Larger datasets and optimized hyperparameters can improve the accuracy of the proposed system. The system proposed in [31] utilizes deep reinforcement learning, feature selection, and optimal hyperparameters to identify anomalous network traffic patterns. In [32], DCNN are employed to enhance the security of industrial IoT systems by training on network traffic patterns. Possible improvements include adding security features or a hybrid approach for better accuracy. Other research aimed to enhance the accuracy of the system by using hyper-optimization algorithms as in [33], The study focused on enhancing the accuracy of network IDS by combining traditional machine learning algorithms (SVM and RF) with deep learning techniques (CNN and RNN). The proposed hybrid model aims to improve the classification effectiveness of intrusion detection systems by leveraging the strengths of both machine learning and deep learning techniques.

Table 1 presents an overview of studies on improving the accuracy of deep learning techniques for intrusion detection systems (IDS). Each study is evaluated based on its methods, dataset, optimization method, accuracy and limitations.

The primary objective of FSBDL is to improve the accuracy and speed of detecting 23 different types of attacks in high-speed networks, compared to traditional deep learning algorithms. With the incorporation of Adam and RMSprop algorithms, the framework's performance is further elevated, making it highly effective in identifying and preventing cyber-attacks.

IV. THE PROPOSED FSBDL FRAMEWORK

The difficulty of this proposal is to perform at the highest accuracy while minimizing processing time, expanding the set of hidden layers in a NN can lead to higher accuracy but also longer processing time. However, we can boost the accuracy of the NN by using hyper parallel optimization techniques. This approach allows us to strike a balance between accuracy and efficiency, making the proposal more viable and practical for real-world applications.

As depicted in Figure 1, the FSBDL framework is built from six layers. The first layer involves data preparation, followed by the second layer that performs feature selection.

TABLE 1. Real-time intrusion detection system: A comprehensive review.

Cat.	Ref. No.	Method	Dataset	Optimizer	Accuracy (%)	Limitation
Feature selection	[24]	Residual learning through ResNet algorithm.	NSL-KDD	Adam	97.20	Time-consuming for large datasets
	[25]	Fusion of statistical importance for FS in DNN-based IDS.	UNSW-NB15	-	98.73	The FS process could be time-consuming for large datasets.
	[26]		NSL-KDD	Swarm	98.30	Highly dependent on the dataset. Result in slower performance. Not scalable.
Filter based FS	[27]	IDS using Tree CNN	CICIDS2017	-	97.90	Not reliable for detecting attacks in complex IoT networks. Lead to misclassifications.
Deep learning	[20]	ANN	NSL-KDD	GA	99.83	GA require significant resources for computing. Not suitable for real-time IDSs.
	[29]		NSL-KDD	Adam	99.10	Not suitable for other datasets. A significant amount of labelled training data is required, Limit its scalability and practicality in real-world applications.
	[30]	DCNN	Vibration signals	Adam	99.80	Require a large amount of labeled training data. The accuracy varies depending on factors such as lighting conditions and image quality
	[31]	Fusing DNN		Adam	97.10	Training and evaluating the model demands a large amount of processing power.
	[32]	DCNN	Acoustic signatures of network	Adam	97.70	Requires specialized hardware and software for signal processing. Not detect all types of network intrusions The accuracy can vary based on the quality of the audio signals and types of attacks detected.
	[33]	Deep reinforcement	Network traffic data	RMSProb	99.30	Need for significant amounts of computation power for training.
	[34]	DCNN	Comm. data	Adam	96.80	Need for significant amounts of computation power for training.
	[35]	CNN	NSL-KDD	Adam	99.20	Require a large amount of data and computational resources. The system may not be as effective in detecting new and unknown types of attacks.

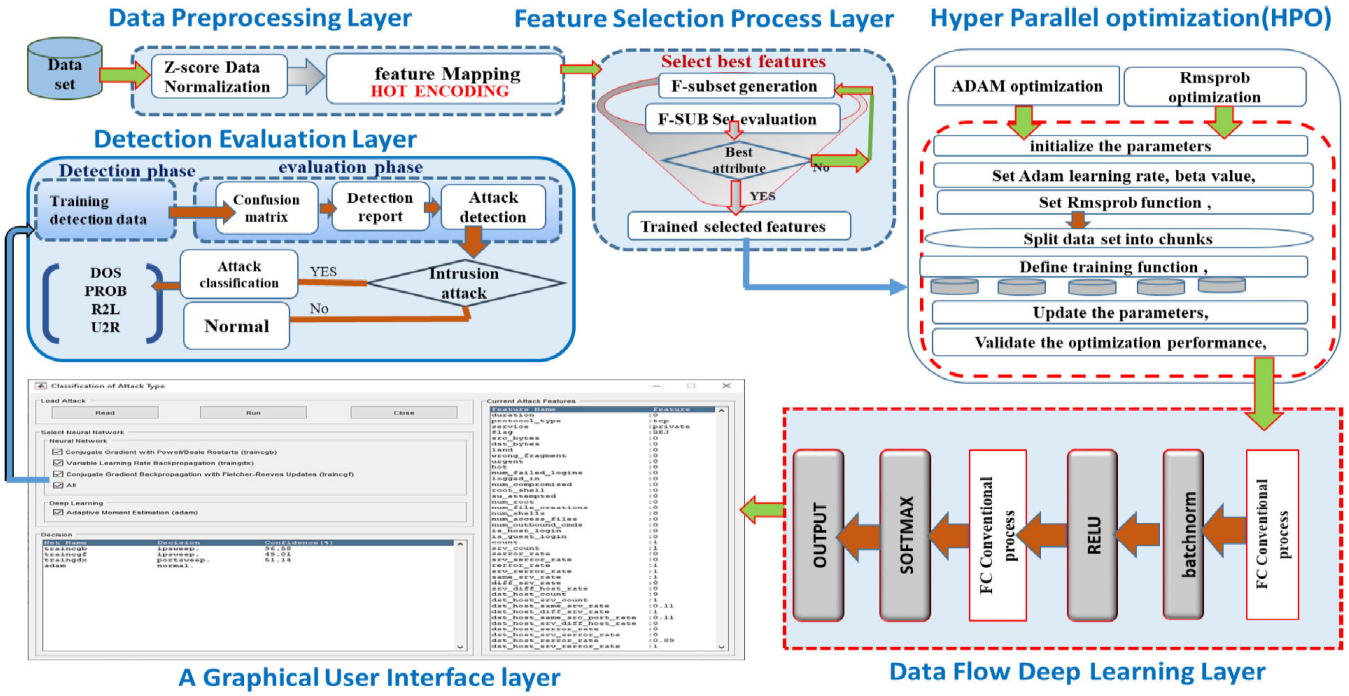

FIGURE 1. The Proposed FSBDL Framework.

The third layer is a hyper-parallel optimization layer aimed at enhancing the accuracy of normalized data. The fourth layer is a data flow deep learning layer that employs advanced deep learning techniques. The fifth layer is dedicated to GUI design. output is then fed into the GUI interface of the IDS. The framework's effectiveness in decision-making for the IDS is evaluated in the final layer, referred to as the detection-evaluation layer.

A. DATA PREPROCESSING LAYER

The NSL-KDD dataset, which is commonly used in IDS evaluations, contains redundant samples due to its continuous data streams. During the pre-processing stage, we first normalize the data and then encode the features into a symbol space to distinguish between raw data.

1) DATA NORMALIZATION

The first step in the data preparation phase involves cleaning the data by removing any redundant information and filling in any missing values. By selecting the most relevant and meaningful features from the data stream, accuracy can be improved and errors in traffic categorization can be reduced. The processing phase involves transforming and standardizing the data, converting categorical values into numerical values as required by the learning algorithm. Z- score normalization is utilized in this phase to decrease variance in large datasets and speed up the training process.

This method was tested on the NSL-KDD dataset to make the training more efficient with cleaned and standardized data

$$z = (x - \mu)/\sigma \tag{1}$$

where z is the z-score normalization, x is the value, μ is the mean of the sample and σ refers to the sample standard deviation [34].

2) FEATURE MAPPING

Feature mapping is a process used to organize categorical data [35], [36]. Hot encoding, also known as feature coding, is a process used to prepare data sets for ML. The initial stage is to pre-process the dataset, followed by identifying the categorical variables that need to be hot-encoded, such as ''flag,'' ''host login,'' ''login,'' ''protocol type,'' ''service,'' and ''ground.'' In step two, a new column is created for each category within each identified categorical variable. Such as, if the ''protocol type'' variable has three categories (TCP, UDP, and ICMP), three new columns would be created.

The NSL-KDD dataset, which includes 41 features for 23 types of attacks divided into persistent and token categories, is used in this process. In step three, the one-hot method is used for symbolic features, resulting in a matrix with regard to the number of instances and records in the total dataset (494022 records), each case having 41 features. However, the large amount of data may slow down system execution, so the features are organized into an array. In step four, symbolic properties are converted into a set of bits representing the feature number, improving accuracy while reducing execution time.

This involves encoding the token features and the 23 types of attacks into numerical form, with each column representing a symbolic property of the database that is labelled and then split into multiple columns. In the final step, the values in the columns are transformed into either 1s or 0s based on their presence. Continuous-type features need to be transformed to a common scale [0, 1] range using the default scaler. This hot encoding process allows for more efficient processing and improved accuracy in identifying and categorizing the data.

B. FEATURE SELECTION PROCESS

To create a feature selection subset from the NSL-KDD dataset, the first step is to perform a comprehensive analysis of the dataset to identify the most important features for the task at hand. This involves evaluating the impact of each characteristic on the relation between the features and the target variable., and assessing each feature's impact on the overall model performance. To determine the ideal set of features, the model is trained and tested on various combinations of features, and the results are compared to identify the most accurate and efficient combination. The selected feature set is then used to train a deep learning model, which is evaluated on a separate test dataset to determine its overall accuracy and identify areas for improvement. This rigorous process ensures that only the most relevant features are utilized in the model, leading to increased accuracy and efficiency in the categorization of traffic within the data stream.

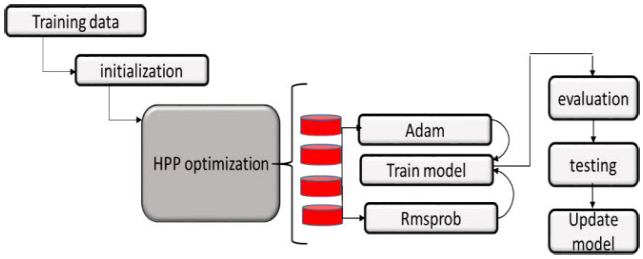

FIGURE 2. Hyper Parallel Optimization Block Diagram.

C. HYPER PARALLEL OPTIMIZATION (HPO)

The proposed framework's novelty lies in the combination of two powerful optimization techniques, Adam and RMSprop, to handle big data stream problems efficiently. We use a combination of optimizers in parallel to select the optimal feature subset for feature extraction. Among them, Adam optimizer adapts the learning rate for each parameter [37], [38] while RMSprop helps prevent oscillations in the optimization process [31], [39].

The FSBDL user subset generation process involves two steps. First, a full search identifies the best combination of features by considering all 2^N candidate subsets for each data set with N features and allowing for the addition or removal of features. In the second step, the evaluated subset is measured using genetic techniques based on distance, information gain, correlation, and similarity features until desired criteria are met. Figure 2 illustrates the process flow.

The use of Hyper Parallel Optimization (HPO) aims to achieve maximum precision and accuracy when training the ID framework. The process starts with inputting the ID dataset, using this technique, the model is trained and then evaluated. The NN model and the Adam and RMSprop optimizers are initialized and configured with respective hyperparameters. The model is trained in parallel using both optimizers by dividing the dataset into two parts

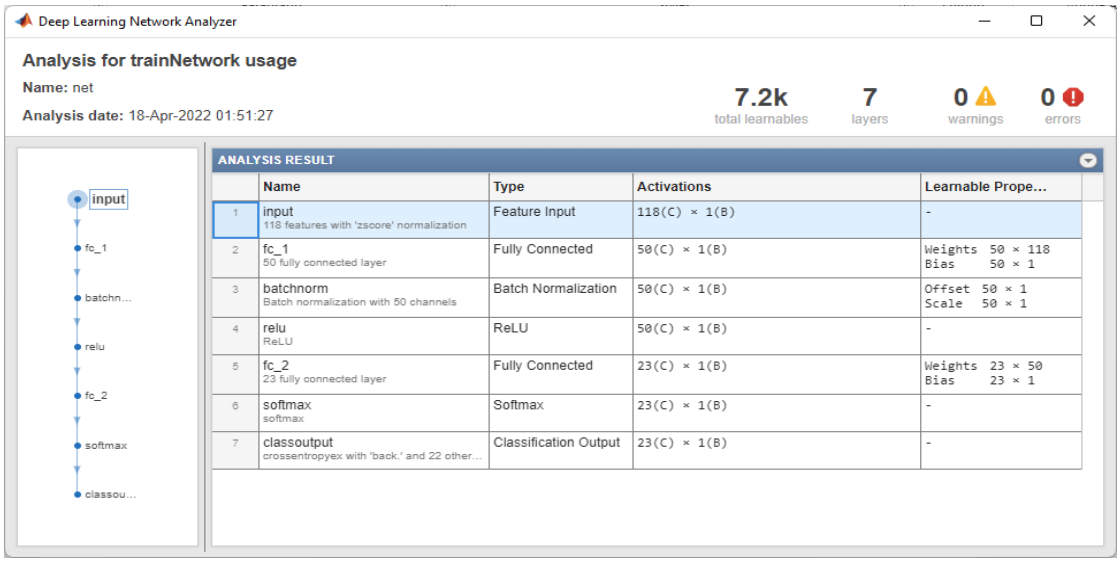

FIGURE 3. Hyper Parallel Optimization Block Diagram.

and simultaneously training with Adam and RMSprop. The trained model is evaluated on a validation dataset to determine performance. The final model with the highest accuracy is selected and evaluated on an unseen test dataset to assess its performance on new data. Further improvements can be made by repeating the process with adjusted hyperparameters. The resulting output is a model with high accuracy, ready to be used in an IDS.

D. DATA FLOW DEEP LEARNING LAYER

The proposed FSBDL is a sequence of layers applied to a deep CNN technique with batch normalization to optimize performance. The arrangement of CNN components is important, and SoftMax and ReLU are used as activation functions. Batch normalization layers are used for a multi-class neural network, allowing independent learning while normalizing output for computation of the loss function.

The proposed CNN network module and its architecture using DL, as well as the data flow DL network analyzer, are shown in Figure 3. The network is symbolized by a Directed Acyclic Graph (DAG) in Figure 3, which helps understand the connectivity between layers, the purpose of activation functions, and the effectiveness of learnable weights and biases. The ''o'' symbol represents the layers of the object, and the arrow ''→-'' shows the link between source and destination points. The Graph layer enables configuration and adjustment of the object's layer plotting. The network consists of 7 layers as shown in the DAG graph object. The framework details are as follows:

1) LAYER 1: DATA INPUT LAYER

It provides 118 random features as an input for the deep neural network framework.

2) LAYER-2 FULLY CONVOLUTION LAYER

The key role of the conventional layer is featuring extraction, the number of channels is 50, the size is 118(c)∗1(B), where B1 is the n1- dimensional bias vector, and the bias vector of 1 layer is used Increase degrees of freedom, reduce the number of connections, and increase processing power.

3) LAYER-3 BATCHNORM

By standardizing the input for each epoch, BN improves the output of the last layer [43]. Our proposed framework simplifies the network by reducing the 50- channel input to a single layer. This not only normalizes the input but also reduces the number of training epochs required. Applying activations to this layer further decreases the number of direct inputs to the previous layer, thereby improving the constancy of the training process and the overall accuracy. With Batch Normalization (BN), the input distribution is converted to a standard distribution of normality with mean 0 and variance 1, which avoids the gradient dispersion problem and increases the size of the gradients. This leads to faster convergence of deep neural networks, thereby speeding up the training process.

4) LAYER 4: RECTIFIED LINEAR UNIT (RELU)

We use an activation function in our proposed data flow layer to clearly differentiate the output as either positive or negative representation [40], [41]. The ReLU layer activation function, with weights 50(c)∗1(B), The threshold operation performed by the system sets any negative values less than zero to zero, as calculated in equation (2):

$$(x) = max(0, x) \tag{2}$$

5) LAYER 5: FULLY-CONNECTED LAYER 2

The input is obtained from the preceding ReLU layer, and the output of the classification layer is thoroughly analyzed. This generates a unit corresponding to 23 traffic classes for multi-class classification.

6) LAYER 6: SOFTMAX

The SoftMax layer is an activation function that maps multiple scalars into a probability distribution with an output range of 0 to 1. The activation function used by the SoftMax layer, SoftMax, converts multiple scalars into a probability distribution between 0 and 1.

7) CLASS OUTPUT

The proposed data flow deep learning algorithm is shown in the following algorithm.

E. THE GRAPHIC USER INTERFACE (GUI) LAYER FOR FULLY ID DEEP LEARNING

The IDS application with a GUI can analyze data behavior and current attack features. It monitors network traffic and analyses received data. An example of the IDS is shown in Figure 4 to demonstrate how the proposed IDS works.

The GUI not only detects real-time data but also provides an accuracy score and identifies the form of attack. In the GUI, a dialog shown in Figure 4 allows the user to run DL with HPO optimization. It displays a list of attacks and consists of three tabs. The first is to ''read'' and upload data, the second is to ''run'' the frame processing layer and make decisions, and the last is to close the tab. The user can select the algorithm type or all methods via checkboxes. For example, in the GUI data case shown in the figure, the user selects the algorithm for the judgment. When the detection score is 100%, NID knows the payload is accurate and displays 41 functions as data cases in the current function box. However, the results may not be completely reliable as the initial IDS results may contain errors.

The approach utilized produces a decision with a specific level of accuracy, assessable through the accuracy percentage. This enables identification of normal or attacked data and determination of attack classification. The labeler displays the predictions generated by leading CNN frameworks as a reference for analysts discerning actual malicious payloads.

F. DETECTION-EVALUATION LAYER

This layer is for detection evaluation. It evaluates the behavior of the data in real time and decides whether the behavior is normal or indicates an intrusion. The system can also detect the type of intrusion, resulting in a high accuracy and low error rate, giving the system high credibility for intrusion detection.

V. EXPERIMENTAL RESULTS AND DISCUSSION

To validate our proposed framework, we conducted ten comparative studies with recent algorithms. These comparisons were made on a computer equipped with Windows 10 operating system, an NVIDIA GeForce RTX 3070 card with 6 GB of graphics memory, and an Intel(R) Core (TM) i7-10700K CPU@3.80GHz featuring 8 cores, as well as 32 GB of RAM. The deep learning model was developed using MATLAB programming language, and Tensor Flow-GPU library with Keras was utilized for computation.

Algorithm 1 ID Deep Learning Using HPO Optimization

Input:
Data stream with a feature input layer
Output: Optimized data set.
x = input_Stream;
t = targetExpanded( );
Start
1. Define network architecture
% Create a network with a feature input layer and specify the number of features. 2. Set hiddenLayerSize = 10;
3. Set miniBatchSize = 1024;
4. Calculate net = CNN (hiddenLayerSize, trainFcn);
% Setup Division of Data for Training, Validation, Testing
5. Net.divideparam.trainratio = 0.7;
6. Net.divideparam.valratio = 0.15;
7. Net.divideparam.testratio = 0.15;
8. 2. Specify training options
9. Calaculate [net,tr] = trainNetwork(x, t, net.Layers, options);
10. [net,tr]=trainNetwork (x, t, net.Layers, options)
11. % Test the Network
12. y = net(x);
13. e = gsubtract(t, y);
14. Calculate performance = perform(net, t, y);
15. tind = vec2ind(t);
16. yind = vec2ind(y);
17. Calculate percentErrors = sum(tind ∼= yind)/numel(tind);
18. Y = round(y)';
19. countAll = 0;
20. For i=1: length(targetExpanded)
21. if Y(i,:) == targetexpanded(i,:)
22. Countall = countall + 1;
23. Calculate accuracy;
24. Length(targetexpanded);
25. Countall = 0;
26. End if
27. End for
28. *numFeatures = size(tbl,2) - 1;*
29. *numClasses = numel(classNames);*
30. *layers = [imageInputLayer([28 28 1]), convolution2dLayer(5,20), reluLayer, maxPooling2dLayer(2,'Stride',2), convolution2dLayer(5,50), reluLayer, maxPooling2dLayer(2,'Stride',2), fullyConnectedLayer(500), reluLayer, fullyConnectedLayer(numClasses), softmaxLayer, classificationLayer];*
31. % Specify Training Options
32. *options = trainingOptions('adam', 'MiniBatchSize', miniBatchSize)*
33. % Train Network
34. *net = trainNetwork(tblTrain, categorical(tblTrain.(labelName)), layers, options);*
35. % Test ALL Network
36. *YPred = classify(net, tbl(:,1:end-1), 'MiniBatchSize', miniBatchSize);*
37. Calculate YTest = tbl{:,labelName};
38. *Accuracy All = sum(YPred == YTest) ∗ 100 / numel(YTest);*
39. For i=1 to numclass
40. *Countclass = zeros (1, numclasses);*
41. *Countcorrectclass = zeros (1, numclasses);*
42. For i = 1: numel (YPred)
43. *Calculate idx = find(YPred(i) == classNames);*
44. *countClass(idx) = countClass(idx) + 1;*
45. IF YPred(i) == YTest(i)
46. *countCorrectClass(idx) = countCorrectClass(idx) + 1;*
47. End For
48. End

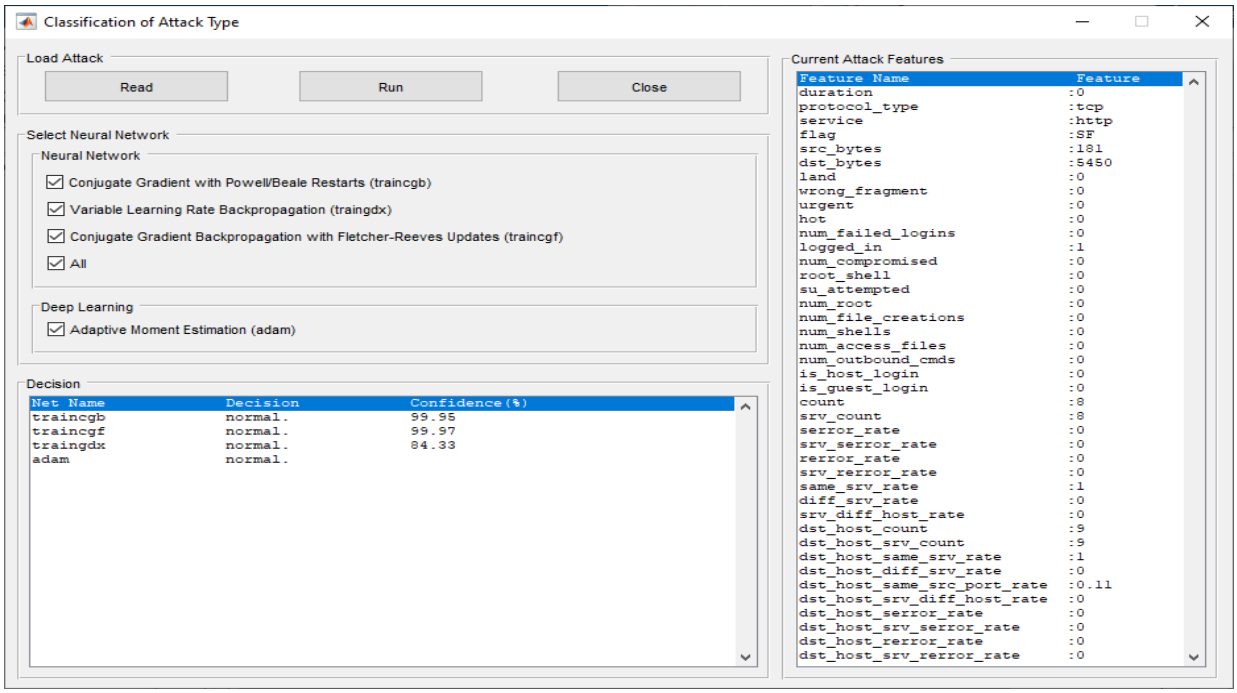

FIGURE 4. Graphical User Interface application for IDS proposed algorithm.

TABLE 2. NSL-KDD classification attacker types.

Class	Types
DOS	*Smurf Land pod teardrop Back nepton*
PROB	*Ipsweep nmap satan*
R2L	*ftp_write guess_passwd imap multihop phf spy warezclient warezmaster.*
U2R	*Rootkit perl loadmodule buffer_overflow portsweep*

A. NSL-KDD DATA SET

The NSL-KDD dataset was chosen as it is widely recognized and widely used as a benchmark for evaluating intrusion detection systems. It contains a large number of different types of network attacks, making it suitable for evaluating various ID approaches [42], [43]. Moreover, it has been used in previous studies and provides a useful point of comparison for our results. Other datasets such as CICID [44], were not considered due to their inappropriateness for our research question and methodology. The use of multiple datasets would have increased the complexity and risk of overfitting the model.

The dataset includes train, validation, and test data, consisting of 23 attack types and common packet data types, divided into four categories: DoS, Probe, U2R, and R2L. Table 2 provides details on the four attack categories and the type of each attack.

Table 3 demonstrates that the NSL-KDD dataset comprises vectors with 41 features and a single class label. Among these, 33 are continuous while the remaining 7 are symbolic. Feature reduction improves classification accuracy and saves computing time. Encoding function understands new function from inputs, input layer is original set, hidden layer helps understand for dimensionality reduction, and output layer represents objective function.

B. PERFORMANCE EVALUATION OF THE FSBDL FRAMEWORK ON STREAMING DATA

This proposal applies the IDS framework to test its efficiency and compare its results with the latest seven recent research studies. The confusion matrices produced by the classification method have two values: true positives and false positives [40]. The NSL-KDD dataset was categorized into (DOS, R2L, U2R, and Prob), and 42 different subsets were generated by adding and removing features. Each subset is evaluated based on a specific criterion to determine the optimal dataset from 494021 instances. The dataset attributes are real-time multivariate, and the DL evaluation metrics are TP, TN, FP, and FN. TP indicates a correct intrusion detection [45], [46], FP means normal behavior was mistakenly classified as an intrusion, TN indicates normal behavior was correctly classified as normal, and FN indicates an undetected attack classified as normal. Common performance metrics, including accuracy, Sensitivity, Specificity and precession are typically used in studies to evaluate the performance of a framework [47], [48].

In this study, we evaluate the framework's performance using four common validation metrics: accuracy, precision, specificity, and sensitivity, outlined in [49]. The calculations for these metrics are as follows:

1) ACCURACY

It describes the ratio of true predictions.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (3)$$

TABLE 3. NSL-KDD feature description.

No	Attribute Name	Description	type
1	duration	Length (number of seconds) of the connection	continuous
2	Protocol-type	Type of the connection protocol	symbolic
3	service	Network service on the destination	symbolic
4	flag	Normal or error status of the connection	symbolic
5	Src-byts	Number of data bytes from source to destination	continuous
6	Dst-byts	Number of data bytes from destination to source	continuous
7	land	1 if the connection is from/to the same host/port; 0 otherwise	symbolic
8	wrong fragment	Number of wrong fragments	continuous
9	urgent	Number of urgent packets	continuous
10	hot	Number of "hot" indicators	continuous
11	num_failed_logins	Number of failed login attempts	continuous
12	Logged in	1 if successfully logged in; 0 otherwise	symbolic
13	num_compromised	Number of "compromised" conditions	continuous
14	su_attempted	1 if "su root" command attempted; 0 otherwise	continuous
15	num_root	Number of root accesses	continuous
16	num_file_creations	Number of file creation operations	continuous
17	Num_shells	Number of shell prompts	continuous
18	num_access_files	Number of operations on access control files	continuous
19	num_outbound_cmds	Number of outbound commands in an ftp operation	continuous
20	dst_host_srv_rerror	% of connections to the current host and specified	continuous
21	_rate	service that has an RST error	continuous
22	is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	symbolic
23	is_guest_login	1 if the login is a "guest" login; 0 otherwise	symbolic
24	count	No. of connections to the same host as the current connection in the past 2 sec.	continuous
25	srv_count	No. of connections to the same service as the current connection in the past 2 sec.	continuous
26	rerror_rate	% of connections that have "SYN" errors (same host connections)	continuous
27	srv_rerror_rate	% of connections that have "REJ" errors (same service connections)	continuous
28	same_srv_rate	% of connections to the same service (same host connections)	continuous
29	diff_srv_rate	% of connections to different hosts (same service connections)	continuous
30	dst_host_count	Count of connections having the same destination host	continuous
31	dst_host_srv_count	Count of connections having the same destination host using the same service	continuous
32	dst_host_same_srv_rate	% of connections having the same destination host using the same service	continuous
33	dst_host_diff_srv_rate	% of different services on the current host	continuous
34	dst_host_same_src_ port_rate	% of connections to the current host having the same src port	continuous
35	dst_host_srv_diff_h ost_rate	% of connections to the same service coming from different hosts	continuous

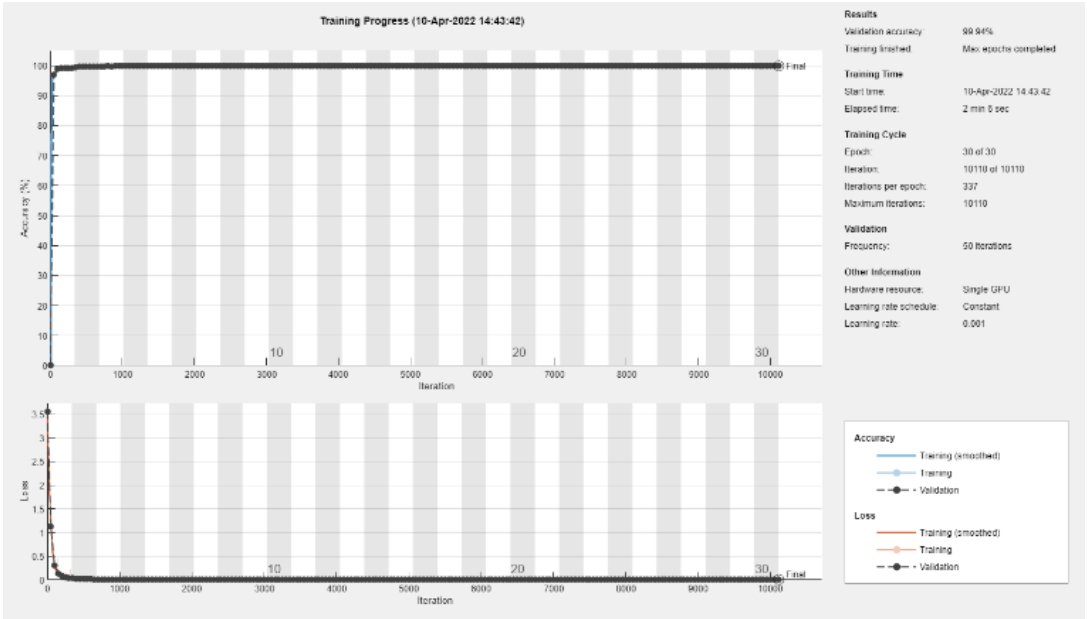

FIGURE 5. Evaluation FSBDL training and testing data in Optimization learning curves after 30 epochs in NSL-KDD dataset.

TABLE 4. FSBDL attack classification learning.

Attacks name	nsl total data		nsl train		nsl validation		nsl test		Accuracy
	size	value	size	value	size	value	size	value	
Back	2203	0.446	1546	0.45	335	0.45	322	0.429	0.99931
land	21	0.004	17	0.00	2	0.00	2	0.003	0.99936
neptune	107202	21.700	75090	21.71	16045	21.65	16066	21.679	0.99936
pod	264	0.053	194	0.06	42	0.06	28	0.038	0.99936
smurf	280790	56.838	196526	56.83	42070	56.77	42193	56.938	0.99936
teardrop	979	0.198	692	0.20	149	0.20	138	0.185	0.99936
ipsweep	1246	0.252	868	0.25	199	0.27	180	0.232	0.99936
nmap	231	0.047	171	0.05	32	0.04	28	0.035	0.99936
satan	1589	0.322	1113	0.32	242	0.33	234	0.314	0.99936
ftp_write	8	0.002	3	0.00	2	0.00	3	0.001	0.99936
guess_passwd	53	0.011	38	0.01	5	0.01	10	0.013	0.99936
imap	12	0.002	12	0.00	0	0.00	0	0.000	0.99936
multihop	9	0.002	2	0.00	4	0.01	1	0.000	0.99936
spy	2	0.000	1	0.00	1	0.00	0	0.000	0.99936
warezclient	1020	0.206	709	0.21	151	0.20	160	0.202	0.99936
warezmaste	20	0.004	11	0.00	1	0.00	8	0.009	0.99936
phf	4	0.001	4	0.00	0	0.00	0	0.0000	0.99936
loadmodule	8	0.002	8	0.00	1	0.00	0	0.000	0.99936
buffer_overflow	30	0.006	20	0.01	8	0.01	2	0.001	0.99936
perl	3	0.001	2	0.00	1	0.00	0	0.001	0.99936
portsweep	1040	0.211	720	0.21	146	0.20	174	0.233	0.99936
rootkit	10	0.002	6	0.00	1	0.00	3	0.001	0.99936
normal	97278	19.691	68061	19.68	14666	19.79	14552	0.000	0.99936
back	2203	0.446	1546	0.45	335	0.45	322	0.429	0.99931
land	21	0.004	17	0.00	2	0.00	2	0.003	0.99936
neptune	107202	21.700	75090	21.71	16045	21.65	16066	21.679	0.99936
pod	264	0.053	194	0.06	42	0.06	28	0.038	0.99936
smurf	280790	56.838	196526	56.83	42070	56.77	42193	56.938	0.99936

2) PRECISION

It represents the percentage of attack connections that are correctly classified as intrusions versus the total number of attack flows:

$$Precision = TP/(TP + FP) \tag{4}$$

3) SENSITIVITY

It is a parameter used to measure the proportion of positive identifications that are correctly classified.

$$Sensitivity = TP/(TP + FN) \tag{5}$$

4) SPECIFICITY

It is a parameter used to measure the proportion of negative identifications that are correctly classified.

$$Specificity = TN/(TN + FP) \tag{6}$$

C. ANALYSIS OF PARALLEL OPTIMIZATION TECHNIQUES AND ON CYBER SECURITY FSBDL FRAMEWORK

The performance of the FSBDL framework was evaluated by optimizing its training parameters and testing the network shown in Figure 5. The validation accuracy, training iteration by epoch, frequency, and learning rate time were confirmed.

The article highlights two challenges faced by classic IDS deep learning algorithms. Firstly, they can be time-consuming due to the flow of data in the dataset. This challenge was addressed by reducing the total number of processes using a large learning rate and utilizing 30 epochs of the Adam-RMSProb parallel optimization algorithm's pre-training process.

Secondly, the NSL-KDD dataset can suffer from overfitting, which can hinder real-time training data classification. To tackle this issue, a GUI was developed to create a real-time dataset solution that can detect intrusions promptly.

The FSBDL framework was developed using the NSL-KDD data set and ReLU activation function on the hidden layer. Binary cross-entropy loss function was used for training. The bottleneck layer's output served as input for the DNN framework. During the generation of output by the initial kernel, categorical cross-entropy function loss was utilized.

The learning rate was set to 0.01, and the CNN had a layer with no significant effect on IDS detection. The Adam optimization algorithm is the best with one, two, or three

TABLE 5. FSBDL framework output multi classification.

Class	All data predicted		Train predicted		Validation predicted		Test predicted	
	Size	ratio	Size	ratio	Size	ratio	Size	ratio
Back	2190	99.41	1535	70.09	335	21.82	318	98.76
land	21	1.00	17	80.95	2	11.76	16065	99.99
neptune	107196	99.99	75085	70.04	16045	21.37	28	100.00
pod	259	98.11	192	74.13	39	20.31	42193	100.00
smurf	280789	100.0	196525	69.99	42070	21.41	137	99.28
teardrop	978	99.90	692	70.76	149	21.53	172	95.56
ipsweep	1226	98.39	859	70.07	195	22.70	26	92.86
nmap	209	90.48	153	73.21	30	19.61	233	99.57
satan	1550	97.55	1082	69.81	236	21.81	1	33.33
ftp_write	2	25.00	16	800.00	1	6.25	10	100.00
guess_passwd	51	96.23	37	72.55	7	18.92	0	0.00
imap	10	83.33	138	1380.00	3	2.17		0.00
multihop	2	22.22	1	50.00	0	0.00	0	0.00
spy	1	50.0	1	100.00	0	0.00	150	93.75
warezclient	966	94.71	671	69.46	145	21.61	7	87.50
warezmaste	17	85.00	9	52.94	1	11.11	0	0.00
phf	4	100.00	4	100.00	0	0.00	0	0.00
loadmodule	1	0.13	1	100.00	0	0.00	1	50.00
buffer_overflow	23	2.21	2	8.70	5	62.50	1	0.00
perl	3	100.00	2	66.67	1	50	173	99.43
portsweep	1030	99.04	711	69.03	146	20.53	1	33.33
rootkit	2	20.00	1	50.00	0	0.00	0	0.00
normal	97191	99.91	67995	69.96	14663	21.56	14615	99.94
back	2190	99.41	1535	70.09	335	21.82	318	98.76
land	21	1.00	17	80.95	2	11.76	16065	99.99
neptune	107196	99.99	75085	70.04	16045	21.37	28	100.00
pod	259	98.11	192	74.13	39	20.31	42193	100.00
smurf	280789	100.0	196525	69.99	42070	21.41	137	99.28

TABLE 6. FSBDL attack classification evaluation Metric.

Class	Accuracy	Precession	Specificity	Sensitivity	F1-measure
back	99.9	98.25	99.9		99.0
land	99.9	100	100	100	100
neptune	99.9	100	100	100	100
pod	99.93	100	100	97.5	98.8
smurf	99.93	100	100	97.5	100
teardrop	99.93	100	100	100	100
ipsweep	99.9	98.4	98.4	98.4	98.4
nmap	99.93	93.75	100	88.2	90.9
satan	99.93	100	100	97.0	98.5
ftp_write	99.9	100	100	100	100
guess_passwd	99.9	100	100	100	100
imap	99.9	100	100	100	100
multihop	99.93	100	100	98.4	98.4
spy	100	-	99.9	-	-
warezclient	99.9	96.3	99.9	92.3	94.2
warezmaste	99.9	100	100	100	100
phf	99.9	-	100	-	-
loadmodule	99.9	100	-	-	-
buffer_overflow	99.9	100	100	100	100
perl	99.9	-	100	-	-
portsweep	99.93	98.7	100	99.3	99.0
rootkit	99.93	100	-	-	-
normal	99.9	98.25	99.9		99.0
back	99.9	100	100	100	100
land	99.9	100	100	100	100
neptune	99.93	100	100	97.5	98.8
pod	99.93	100	100	97.5	100
smurf	99.93	100	100	100	100

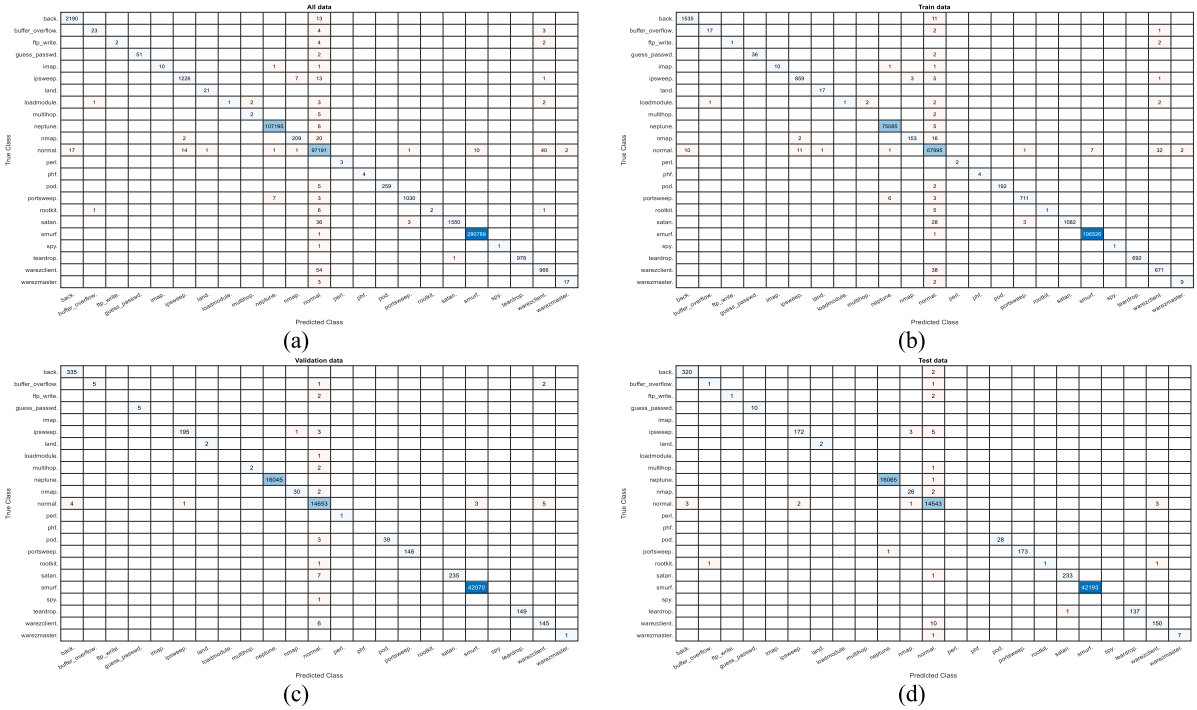

FIGURE 6. IDS implementation on FSBDL deep learning Confusion matrix (a) CM for all data (b) CM for train data (c) CM for validation (d) CM for test data.

hidden layers. 0.01 is the best learning rate for NSL-KDD. The batch-norm size should be 1024, and the learning rate should be 0.01. The detailed results of NSL-KDD for DL are shown in Table 4 in several cases, such as training, validation, testing, and results of all data. The values in table show that certain types of attackers are difficult to predict, such as R2L, including FTP write, multihop, spy, and imap, and U2R, including load modules, (rootkit, buffer overflow, and Perl) attacks. Results are predictable traces for each attack class (ftp-write, multihop, spy, imap, load module, rootkit, buffer-overflow, and Perl attacks) After the learning process, the small amount of training data from the NSL KDD dataset caused (0.002, 0.002, 0.000, 0.002, 0.002, 0.0020.006, 0.001).

Table 5 explains the prediction process metrics for each attack category to reveal the capabilities of the framework. The detection rate was only 60.87 attacks were detected as intruders, and the resulting prediction rate for each attack category (ftp-write, multihop, spy, imap, load module, rootkit, buffer-overflow, and Perl attacks) tracked, 0.25, 22. 2, 50, 83.33, 0.13, 20.00, 76.67, and 100.00. The ratio refers to the high performance of the detection type according to each total amount. Because modifying attacks to make them indistinguishable from normal behavior is intentionally difficult, it's possible to detect modified attacks by comparing packet activity against normal behavior. These are 8 out of 46 modified attacks against 46 normal labels during testing. During data testing, 11 modified attacks against 46 labels were detectable. Its implication means that the IDS framework cannot generalize to U2R attacks. For the R2L attack class, the predicted value from the training data is almost 71.50

TABLE 7. Comparison of the Proposed Framework and State-of-the-Art Stream Data IDS Method.

No.	Method	2	3	4	5
1	CNN+Adam cross entropy [39]	-	0.9861	0.8723	0.9257
2	CNN-RNN[16]	0.9690	0.997	0.9260	0.9600
3	Adam real-time [17]	0.9807	0.9706	0.9902	0.9813
4	SAE[50]-Lightwhaight DS	0.8330	-	0.9450	0.7480
5	Hybrid (CAE+DNN)[47]	0.9129	0.9208	0.9064	0.9135
6	Adam-XGBoost–DNN [51]	0.976	0.97	-	0.97
7	Adam Multiclass Classifier real-time [52]	0.785	0.810	0.785	0.765
8	Adam [53]	0.980	0.999	0.988	0.9938
9	DNN [54]	0.994	0.997	0.979	0.9880
10	DNN [25]	0.998	0.999	0.988	0.9930
11	Proposed FSBDL	**0.9994**	**0.9993**	**0.9993**	**0.9993**

(1611 out of 2253). However, for the test data, the NSL-KDD test, train, and validation TPRs are only 20 and 48.25. Respectively. Furthermore, the imbalanced factory is.80 for all training data. It can be seen more clearly in the confusion matrix that the R2L attack is detected as normal packets on both the training and test data. Even with test data, a normal packet represents approximately 60 of the 2,754 R2L attacks. Confusion matrix (CM) in the Figure 6 (a. HPO Confusion Matrix, b. Confusion Matrix for all predicted data, Confusion Matrix Train, Test, and Validation appears the details.

Figure 6 displays the attacker's activity after data processing training, validation, and testing phases. The attacker's behavior is observed in Figures 6a-d, with Figure 5a showing their normal behavior during data processing. Tables 6 demonstrate the performance of a specific IDS implementation using the FSBDL framework. The Table shows an almost 100% detection rate for various types of attacks, including perl, phf, and s/m, during training, validation, and testing stages. However, some subclasses of attack may not be recognized by the framework, resulting in slightly lower detection rates during testing. It is believed that this framework may not be able to detect every attack within its classes.

Although an attack may belong to a single class, its pattern may still exhibit significant variations based on other attacks in the same class. Without any specific learning or training for that particular attack, it is unlikely that the framework will be able to detect it. Anomaly-based IDS systems detect unusual behaviour in packets and compare that behaviour to that of a normal packet.

D. COMPARATIVE ANALYSIS OF FSBDL AND STATE-OF-THE-ART CYBER SECURITY FRAMEWORKS BASED ON DEEP LEARNING ALGORITHM

The effectiveness of anomaly-based IDS systems is limited when it comes to detecting stealthy attacks as these types of attacks do not display unusual behavior. Stealthy attacks are executed in a way that avoids detection by modifying the normal protocol, including the execution, action and cleaning processes. To avoid drawing attention, attackers spread out the attack over multiple sessions and an extended period, utilizing basic services to prevent unusual commands from being detected as normal packages. The host-based IDS system in use employs log data to identify abnormal network activity.

In terms of accuracy, precision, sensitivity, and other metrics, we compared the results of our FSBDL framework presented in section v.2 with ten recent studies that utilized various DL techniques on the NSL-KDD dataset. As demonstrated in Table 7, our approach outperforms the other studies.

VI. CONCLUSION AND FUTURE WORK

In the context of deep learning applications, achieving high accuracy and fast running times can be challenging due to the relationship between accuracy and the number of hidden layers and implementation time. To address this issue, a proposal suggests sacrificing one objective for the other by using three phases.

Phase one involves feature selection using an HPO optimizer to reduce the data volume. Phase two focuses on data flow using a deep learning layer with CNN, batch normalization, and soft-max activation. Finally, phase three introduces a GUI that continuously monitors accuracy and time to achieve the best combination of the two.

Additionally, the proposal introduces a GUI module designed to allow the system to monitor network activity and record what occurred during the training algorithm selection. The proposed framework was tested against ten recent studies and showed superior accuracy, precision, specificity, sensitivity, and F1 scores of 99.93%.

To further improve the effectiveness of intrusion detection systems, future work can investigate the explainability and interpretability of deep learning models, evaluate performance optimization techniques on unbalanced attack datasets, and develop algorithms capable of detecting emergency changes to network data and anomalies.

The proposed framework has numerous applications, including drone identification and other areas where data security is crucial. To further improve accuracy and efficiency, transfer learning techniques will be explored, along with integrating the framework with parallel optimization algorithms like Adam and RMSprop Future research in these areas is expected to result in substantial breakthroughs in the realm of ID and information security, including protecting sensitive customer information, preventing financial fraud, and securing critical infrastructure.

REFERENCES

[1] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Streaming big data analysis for real-time sentiment based targeted advertising," *Int. J. Electr. Comput. Eng. (IJECE)*, vol. 7, no. 1, p. 402, Feb. 2017.

[2] R. Fontugne, J. Mazel, and K. Fukuda, "Hashdoop: A MapReduce framework for network anomaly detection," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2014, pp. 494–499.

[3] S. K. Sahu, D. P. Mohapatra, J. K. Rout, K. S. Sahoo, Q.-V. Pham, and N.-N. Dao, "A LSTM-FCNN based multi-class intrusion detection using scalable framework," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107720.

[4] Q. Liu, D. Wang, Y. Jia, S. Luo, and C. Wang, "A multi-task based deep learning approach for intrusion detection," *Knowl.-Based Syst.*, vol. 238, Feb. 2022, Art. no. 107852.

[5] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Inf. Sci.*, vol. 513, pp. 386–396, Mar. 2020.

[6] C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Inf. Sci.*, vol. 502, pp. 279–296, Oct. 2019.

[7] I. H. Sarker, "Deep cybersecurity: A comprehensive overview from neural network and deep learning perspective," *Social Netw. Comput. Sci.*, vol. 2, no. 3, p. 154, May 2021.

[8] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug Discovery Today*, vol. 23, no. 6, pp. 1241–1250, Jun. 2018.

[9] S. A. Ajagbe, K. A. Amuda, M. A. Oladipupo, O. F. Afe, and K. I. Okesola, "Multi-classification of Alzheimer disease on magnetic resonance images (MRI) using deep convolutional neural network (DCNN) approaches," *Int. J. Adv. Comput. Res.*, vol. 11, no. 53, pp. 51–60, Mar. 2021.

[10] S. A. Ajagbe, O. A. Oki, M. A. Oladipupo, and A. Nwanakwaugwu, "Investigating the efficiency of deep learning models in bioinspired object detection," in *Proc. Int. Conf. Electr., Comput. Energy Technol. (ICECET)*, Jul. 2022, pp. 1–6.

[11] H. Song, Z. Jiang, A. Men, and B. Yang, "A hybrid semi-supervised anomaly detection model for high-dimensional data," *Comput. Intell. Neurosci.*, vol. 2017, pp. 1–9, Nov. 2017.

[12] W. M. Shaban, A. H. Rabie, A. I. Saleh, and M. A. Abo-Elsoud, "A new COVID-19 patients detection strategy (CPDS) based on hybrid feature selection and enhanced KNN classifier," *Knowl.-Based Syst.*, vol. 205, Oct. 2020, Art. no. 106270.

[13] L. M. Q. Abualigah, *Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering*, vol. 816. Berlin, Germany: Springer, 2019, pp. 1–165.

[14] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102419.

[15] J. Yu, X. Ye, and H. Li, "A high precision intrusion detection system for network security communication based on multi-scale convolutional neural network," *Future Gener. Comput. Syst.*, vol. 129, pp. 399–406, Apr. 2022.
[16] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.
[17] A. Kim, M. Park, and D. H. Lee, "AI-IDS: Application of deep learning to real-time web intrusion detection," *IEEE Access*, vol. 8, pp. 70245–70261, 2020.
[18] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-stage optimized machine learning framework for network intrusion detection," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1803–1816, Jun. 2021.
[19] S. Mishra, R. Sachan, and D. Rajpal, "Deep convolutional neural network based detection system for real-time corn plant disease recognition," *Proc. Comput. Sci.*, vol. 167, pp. 2003–2010, Jan. 2020.
[20] M. Choraś and M. Pawlicki, "Intrusion detection approach based on optimised artificial neural network," *Neurocomputing*, vol. 452, pp. 705–715, Sep. 2021.
[21] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.
[22] P. Chaudhary, B. Gupta, and A. K. Singh, "Implementing attack detection system using filter-based feature selection methods for fog-enabled IoT networks," *Telecommun. Syst.*, vol. 81, no. 1, pp. 23–39, Sep. 2022.
[23] J. Xie, Z. Song, Y. Li, Y. Zhang, H. Yu, J. Zhan, Z. Ma, Y. Qiao, J. Zhang, and J. Guo, "A survey on machine learning-based mobile big data analysis: Challenges and applications," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–19, Aug. 2018.
[24] A. Shaikh and P. Gupta, "Real-time intrusion detection based on residual learning through ResNet algorithm," *Int. J. Syst. Assurance Eng. Manage.*, vol. 13, pp. 1–15, Jan. 2022.
[25] A. Thakkar and R. Lohiya, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Inf. Fusion*, vol. 90, pp. 353–363, Feb. 2023.
[26] A. Nazir and R. A. Khan, "A novel combinatorial optimization based feature selection method for network intrusion detection," *Comput. Secur.*, vol. 102, Mar. 2021, Art. no. 102164.
[27] R. V. Mendonça, A. A. M. Teodoro, R. L. Rosa, M. Saadi, D. C. Melgarejo, P. H. J. Nardelli, and D. Z. Rodríguez, "Intrusion detection system based on fast hierarchical deep convolutional neural network," *IEEE Access*, vol. 9, pp. 61024–61034, 2021.
[28] P. Li, X. Jia, J. Feng, F. Zhu, M. Miller, L.-Y. Chen, and J. Lee, "A novel scalable method for machine degradation assessment using deep convolutional neural network," *Measurement*, vol. 151, Feb. 2020, Art. no. 107106.
[29] Y. Xu, "Intrusion detection based on fusing deep neural networks and transfer learning," in *Proc. Int. Forum Digit. TV Wireless Multimedia Commun.* Shanghai, China: Springer, Sep. 2019, pp. 212–223.
[30] M. Y. Aldarwbi, A. H. Lashkari, and A. A. Ghorbani, "The sound of intrusion: A novel network intrusion detection system," *Comput. Electr. Eng.*, vol. 104, Dec. 2022, Art. no. 108455.
[31] S. Bakhshad, V. Ponnusamy, R. Annur, M. Waqasyz, H. Alasmary, and S. Tux, "Deep reinforcement learning based intrusion detection system with feature selections method and optimal hyper-parameter in IoT environment," in *Proc. Int. Conf. Comput., Inf. Telecommun. Syst. (CITS)*, Jul. 2022, pp. 1–7.
[32] H. Mankodiya, N. K. Jadav, S. Tanwar, and R. Gupta, "Deep learning-based secure Machine-to-Machine communication in edge-enabled industrial IoT," in *Proc. Int. Conf. Comput., Commun., Intell. Syst. (ICCCIS)*, Nov. 2022, pp. 48–53.
[33] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *J. Inf. Secur. Appl.*, vol. 58, May 2021, Art. no. 102804.
[34] K. A. Al-Utaibi and E.-S.-M. El-Alfy, "Intrusion detection taxonomy and data preprocessing mechanisms," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1369–1383, Mar. 2018.
[35] S. M. El-Ghamrawy, A. I. El-Desouky, and M. Sherief, "Dynamic ontology mapping for communication in distributed multi-agent intelligent system," in *Proc. Int. Conf. Netw. Media Converg.*, Mar. 2009, pp. 103–108.
[36] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.
[37] A. Sarkar, H. S. Sharma, and M. M. Singh, "A supervised machine learning-based solution for efficient network intrusion detection using ensemble learning based on hyperparameter optimization," *Int. J. Inf. Technol.*, vol. 15, pp. 423–434, Oct. 2022.
[38] A. M. Mahfouz, D. Venugopal, and S. G. Shiva, "Comparative analysis of ML classifiers for network intrusion detection," in *Proc. 4th Int. Congr. Inf. Commun. Technol., (ICICT)*, vol. 2. London, U.K.: Springer, 2020, pp. 193–207.
[39] W. Zheng, "Intrusion detection based on convolutional neural network," in *Proc. Int. Conf. Comput. Eng. Appl. (ICCEA)*, Mar. 2020, pp. 273–277.
[40] S. Kabir, S. Sakib, M. A. Hossain, S. Islam, and M. I. Hossain, "A convolutional neural network based model with improved activation function and optimizer for effective intrusion detection and classification," in *Proc. Int. Conf. Advance Comput. Innov. Technol. Eng. (ICACITE)*, Mar. 2021, pp. 373–378.
[41] R. Lohiya and A. Thakkar, "Intrusion detection using deep neural network with antirectifier layer," in *Applied Soft Computing and Communication Networks*. Cham, Switzerland: Springer, 2021, pp. 89–105.
[42] R. Pecori, A. Tayebi, A. Vannucci, and L. Veltri, "IoT attack detection with deep learning analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.
[43] S. Elghamrawy, "An H_2O's deep learning-inspired model based on big data analytics for coronavirus disease (COVID-19) diagnosis," in *Big Data Analytics and Artificial Intelligence Against COVID-19: Innovation Vision and Approach*. Cham, Switzerland: Springer, 2020, pp. 263–279.
[44] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107390.
[45] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.
[46] A. Rashid, M. J. Siddique, and S. M. Ahmed, "Machine and deep learning based comparative analysis using hybrid approaches for intrusion detection system," in *Proc. 3rd Int. Conf. Advancements Comput. Sci. (ICACS)*, Feb. 2020, pp. 1–9.
[47] V. Dutta, M. Choraś, R. Kozik, and M. Pawlicki, "Hybrid model for improving the classification effectiveness of network intrusion detection," in *Proc. 13th Int. Conf. Comput. Intell. Secur. Inf. Syst. (CISIS)*. Cham, Switzerland: Springer, Dec. 2021, pp. 405–414.
[48] S. M. EL-Ghamrawy and A. I. Eldesouky, "An agent decision support module based on granular rough model," *Int. J. Inf. Technol. Decis. Making*, vol. 11, no. 4, pp. 793–820, Jul. 2012.
[49] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020, *arXiv:2010.16061*.
[50] M. Lin, B. Zhao, and Q. Xin, "ERID: A deep learning-based approach towards efficient real-time intrusion detection for IoT," in *Proc. IEEE 8th Int. Conf. Commun. Netw. (ComNet)*, Oct. 2020, pp. 1–7.
[51] P. Devan and N. Khare, "An efficient XGBoost–DNN-based classification model for network intrusion detection system," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12499–12514, Aug. 2020.
[52] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[53] S. Alzughaibi and S. El Khediri, "A cloud intrusion detection systems based on DNN using backpropagation and PSO on the CSE-CIC-IDS2018 dataset," *Appl. Sci.*, vol. 13, no. 4, p. 2276, Feb. 2023.
[54] L. Li, R. Ahmad, W. Tsai, and A. K. Sharma, "A feature selection based DNN for intrusion detection system," in *Proc. 15th Int. Conf. Ubiquitous Inf. Manage. Commun. (IMCOM)*, Jan. 2021, pp. 1–8.