GMR
The Original

MODELLING A DEEP LEARNING BASED HYBRIDIZED ENSEMBLE LEARNING APPROACH FOR INTRUSION DETECTION IN INDUSTRIAL ENVIRONMENT

K.Arun Prasad [1], Dr. G. Pattabirani[2], Dr. K. Sundaramoorthy[3]

[1] Research Scholar, Department of Information Technology, Annamalai University
[2]Assistant Professor, Department of Information Technology, Annamalai University
[3]Professor, Department of Information Technology, Jerusalem College of Engineering
Mail Id: Arunprasadchamy@gmail.com[2], Prvijayaraja@gmail.com[2], ksundaramoorthy1999@gmail.com[3]

ABSTRACT
A network intrusion detection system (NIDS) is one of the primary approaches for preventing cyber-attacks. Deep learning (DL) has shown promising results for intrusion detection, yet the effect of feature fusion on improving accuracy and generalization in NIDS remains underexplored. This study proposes novel deep learning frameworks employing feature fusion strategies to enhance the reliability of multi-class classification in NIDS. This work introduces three distinct approaches based on Ensemble Convolutional Long Short-Term Memory (Ensemble C-LSTM) models that integrate multiple feature types to address class imbalance and capture inter-feature correlations effectively. Publicly available datasets, UNSW-NB15 and NSL-KDD, were used to evaluate the proposed models against existing algorithms. Experimental results demonstrate that the late-ensemble and late-fusion approaches achieved superior generalization, with consistent performance on validation and test sets, while minimizing overfitting. Compared to conventional DL and modern algorithms, the proposed frameworks deliver improved robustness in handling class imbalance and complex classification tasks. These results indicate that feature-fusion-based deep learning models offer a promising pathway toward building reliable intrusion detection systems capable of addressing evolving cyber threats.

KEYWORDS: accuracy, attack, ensemble, learning, prediction

1. INTRODUCTION

The Gartner research "Market Guide for AIOps Platforms" states that people cannot wait for the exponential growth of event information to yield insights. Secure information technology operations require automation and machine learning (ML) support. The study also observes that because correlation regulations need to be updated quickly, rule-based correlation between events has been replaced with AI-based correlation [1]. Actually, according to the websites of any contemporary cyber security supplier, emerging threats like polymorphic viruses can defeat classic signature-based responses, which is why more adaptive solutions utilizing machine learning are required. The Network Intrusion Detection System (NIDS) is one of the primary techniques used to protect against cyber-attacks [2]. A NIDS is an equipment that monitors network traffic from a cyber-security perspective. It is situated on an internet connection in a key spot, usually within a perimeter firewall. When it recognizes events of possible cyber security significance in a continuous flow of packets, it notifies administrators, Security Information and Event Management (SIEM) networks, or other equipment [3]. The tools like Network Intrusion Prevention or Detection and Prevention Systems (NIPS or NIDPS) are utilized to identify potentially dangerous activities and take necessary steps. Instead of relying on signatures, anomaly recognition strategies make the (fair) presumption that malicious activity will manifest itself through modifications to the system's behavior under observation [4]. As NIDSs deal with various data techniques [5] – [10], they must effectively manage this diversity. Several deep multimodal neural network architectures that can combine signals from different sensors or features from different modalities are investigated by [11]. Their research thoroughly analyzes the effects of adding obtained multimodal visualizations to DL models. Deep learning has great potential in solving complex ML problems by utilizing the availability of various data modalities. To effectively manage diverse data modalities and improve multi-categorization efficiency [12], an investigation is needed concerning creating innovative deep learning frameworks and feature fusion techniques suited explicitly to NIDSs. Our investigation fills this gap by introducing novel deep learning design concepts using unique feature fusion processes designed to improve NIDS performance [13]. Similar to approaches used in multimodal learning models, the proposed techniques include layers that integrate different features and enable an understanding of their interdependencies. Notably, the feature fusion algorithms were carefully applied to support the DL algorithm's ability to identify the connections among inputs at various training phases and levels [14] – [15]. Existing research often relies on public datasets that are synthetic, limited

in attack variety, or fail to capture accurate protocol-level semantics, such as Modbus, DNP3, and IEC 104. Very few datasets reflect modern, converged IT/OT environments that include background noise, routine maintenance traffic, and realistic attack timelines. Moreover, most models are trained and evaluated on a single dataset or simulated plant, with little examination of how well they transfer across sites featuring different device types, sampling rates, or operational workflows. Industrial control systems (ICS) and operational technology (OT) environments including SCADA networks, PLC systems, and smart factories are increasingly interconnected with enterprise IT and cloud infrastructures, making them vulnerable to sophisticated cyber-attacks. Traditional intrusion detection systems (IDS) in these settings, which rely on signature-based methods or rule engines, often fail to identify novel, stealthy, or multi-stage attacks and tend to generate high false-positive rates when applied to OT traffic. DL approaches can capture subtle anomalies through learned representations, while ensemble learning can enhance model robustness and generalization. However, standalone DL models can be computationally intensive, lack interpretability, and are often sensitive to adversarial manipulations and concept drift. To address these challenges, this study explores a hybridized ensemble IDS that integrates lightweight deep learning feature extractors with complementary machine-learning classifiers and ensemble fusion techniques tailored to the unique constraints and threat landscape of industrial environments. The goal is to achieve high detection accuracy for both known and emerging attack types, maintain a low false-positive rate suitable for operational deployment, enable real-time or near real-time inference on resource-limited edge devices, and provide interpretable alerts to support effective operator decision-making. This work introduces a protocol-aware, hybridized ensemble intrusion detection system tailored for industrial control systems. Unlike conventional CNN- or LSTM-based detectors that rely on single-model representations and often require heavy resources, our approach fuses compact deep feature extractors with complementary classical classifiers and a calibrated meta-learner, explicitly optimizing for edge deployment, adversarial resilience, and concept-drift adaptation. We further map detections back to protocol-level semantics to provide concise, operator-ready explanations. Extensive cross-site and edge-centric evaluations demonstrate improved per-class recall, substantially reduced false alarms in operational settings, and lower inference latency compared to standard CNN and LSTM baselines. Therefore, more research and analysis are necessary to determine the effectiveness and superiority of our suggested models over current methods in the field of NIDS. This work use the deep, fully linked structure as an example to demonstrate the effectiveness of our recommended solutions. The following depicts the summary of the proposed system:

1. Combining different feature types to create a unified characteristic vector can be used as an feed in for a system with all its connections. Three steps are used: first, a processing layer changes the signals, and then the transformed signals are combined and applied into a fully linked network.
2. Enabling learning of typical local and hierarchical properties through signal fusion inside a unique network. Utilizing ensemble learning known as Ensemble Convolutional Long Short Term Memory (Ensemble C-LSTM) to take advantage of ensemble and hybridized models. Combining the two systems' high-level representations to create a single feature vector.
3. The work is drafted in the following manner: section 2 gives a deeper analysis of various ideas provided by the researchers. The proposed methodology is explained in section 3. The numerical results are provided with graphical outcomes in section 4, and the conclusion is summarized in section 5.

2. RELATED WORKS

Many openly accessible datasets, such as KDDCup99, DARPA_1998, NSL-KDD, and UNSWNB15, have been created to support the growing interest in using ML approaches to enhance vulnerability identification capabilities [16] – [17]. The author in [18] recommends combining supervised and unsupervised ML approaches with k-means and random forest on an intrusion recognition dataset. According to the study, this hybrid strategy outperforms standard methods regarding results. To identify the best classifier for malware detection [19] the author evaluated many supervised methods of ML, including the Random Forest Classifier (RFC), using the NSL-KDD repository. Performance parameters like F1-score, precision, and recall were compared [20]. The experiment's findings for the NSL-KDD database and feature set showed that the RF classification algorithm performed better than other quantitative ML classifiers. A Random Effects Logistic Regression (RELR) model was suggested by Reference. [21] as a way to predict the finding of anomalies. It included a binding feature extraction stage based on fixed-effects logistic regression (FELR) and used a random-effects framework to prepare for uncertainty and network environment factors. Using combined k-means and correlational modeling for feature extraction, and finally, NB and DT for categorization, a study was conducted on the UNSW-NB15 dataset to identify the varieties of cyber-attacks that have taken place [22]. This hybrid feature selection process significantly improved the reliability of the Naive Bayes classifier; nevertheless, the decision trees' performance was comparable with and without choosing features. Worms, Shellcode, and BackDoor were the more unusual threats the suggested method could detect. Reference [23] offers a new method for modeling program behavior in detection systems for intrusion that relies on the k-nearest Neighbor classifier methodology. It does not demand different profiles of short system call patterns for various applications, unlike earlier methods that employ short system call patterns [24] – [25]. The results also show that achieving a low false positive rate is possible. Text classification techniques appear well suited for security monitoring applications, while this result could not hold for more complex datasets.

Deep learning's capacity to extract high-level characteristics from big datasets with intricate distributions has also led to its remarkable performance in various cyber-security-related activities. By running the information across multiple hidden layers, structured representations of features are automatically learned, eliminating the

requirement for feature construction. As a result, one does not require knowledge of the subject to understand the relevance of new data [26]. Here, eight distinct kinds of cyber threats were studied using various deep learning approaches, such as identifying malware and identifying anomalies. Two methods are used to classify the threats in the network data [27]: a soft-max regression framework employed without characteristic learning and a DL technique known as "self-taught learning," which first employs a sparse auto_encoding device to unlabeled feature before using a NN classification algorithm on the labeled feature. The "self-taught learning" strategy performed better than the alternative. To increase the rate of identification and reliability in the KDDCup99 information set, a deep learning structure was created utilizing a deep belief network (DBN) and a constrained Boltzmann machine. In contrast to DBN, CNN was employed in [28] to detect several attack categories and was created to extract characteristics from images. Considering a deep learning method for recognizing intrusions, a non-symmetric deep auto-encoder (NDAE) for unsupervised representation learning was suggested in [29] and tested on the KDD_Cup99 data repository [30]. To overcome the shortcomings of the existing IDS on the IoT, suggest a novel DL approach that uses two-way LSTM and a symmetric logarithmic loss function. This approach performs excellently on standard datasets like UNSW-NB15 and NSL-KDD, but it only applies to binary classification tasks. Despite the noteworthy achievement of deep learning and traditional machine learning in detecting network intrusions, it is clear from reviewing current developments in associated models that feature fusion techniques still need to be explicitly investigated, and feature selection is still a difficult challenge [30].

The existing literature consistently highlights several unresolved challenges: (a) a lack of realistic, protocol-rich datasets and insufficient cross-site validation; (b) reliance on single deep-learning models that are computationally expensive and highly vulnerable to concept drift and adversarial manipulation; (c) minimal consideration of protocol semantics when designing features and model architectures; (d) limited focus on producing interpretable outputs suitable for human operators; and (e) incomplete evaluation frameworks that overlook operational metrics such as inference latency, memory usage, and false-alarm rates. Existing research often relies on public datasets that are synthetic, limited in attack variety, or fail to capture accurate protocol-level semantics, such as Modbus, DNP3, and IEC 104. Very few datasets reflect modern, converged IT/OT environments that include background noise, routine maintenance traffic, and realistic attack timelines. Moreover, most models are trained and evaluated on a single dataset or simulated plant, with little examination of how well they transfer across sites featuring different device types, sampling rates, or operational workflows. Industrial control systems (ICS) and operational technology (OT) environments including SCADA networks, PLC systems, and smart factories are increasingly interconnected with enterprise IT and cloud infrastructures, making them vulnerable to sophisticated cyber-attacks. The proposed protocol-aware, hybrid ensemble approach integrating lightweight deep-learning feature extractors with classical machine-learning classifiers, mechanisms for drift and adversarial robustness, edge-efficient optimization strategies, cross-site or federated training, and protocol-grounded explanations directly addresses these gaps. This design aligns well with the needs identified in recent surveys and empirical studies and represents a significant advancement over existing methods.

3. METHODOLOGY

This section describes our suggested deep-learning methods for identifying network intrusions. After summarizing the design of the three suggested DL algorithms and discussing their initial training procedures, the section begins with an overview of the datasets utilized in this work as in Fig 1.

3.1. Dataset

Both synthetic intrusion behaviors and genuine, everyday network traffic are present in UNSW-NB15. A network audit device called Argus was used to generate information flow (a set of data packets that were transmitted and received between the source and the destination), and a network congestion analyzer to generate integration-based information gathered from packet data that represented the mix of typical and unusual traffic. After identifying the network topologies and flows, 35 packet-based and flow-based features were obtained. Twelve more features were then retrieved using custom techniques. The KDD-Cup99 malware detection database has been modified to become NSL-KDD. When classifiers are trained on KDD-Cup99, issues like duplicate data that may cause significant bias are addressed using NSLKDD. Data collected throughout the DARPA 1998 intrusion detection system evaluation program served as the basis for creating the dataset.

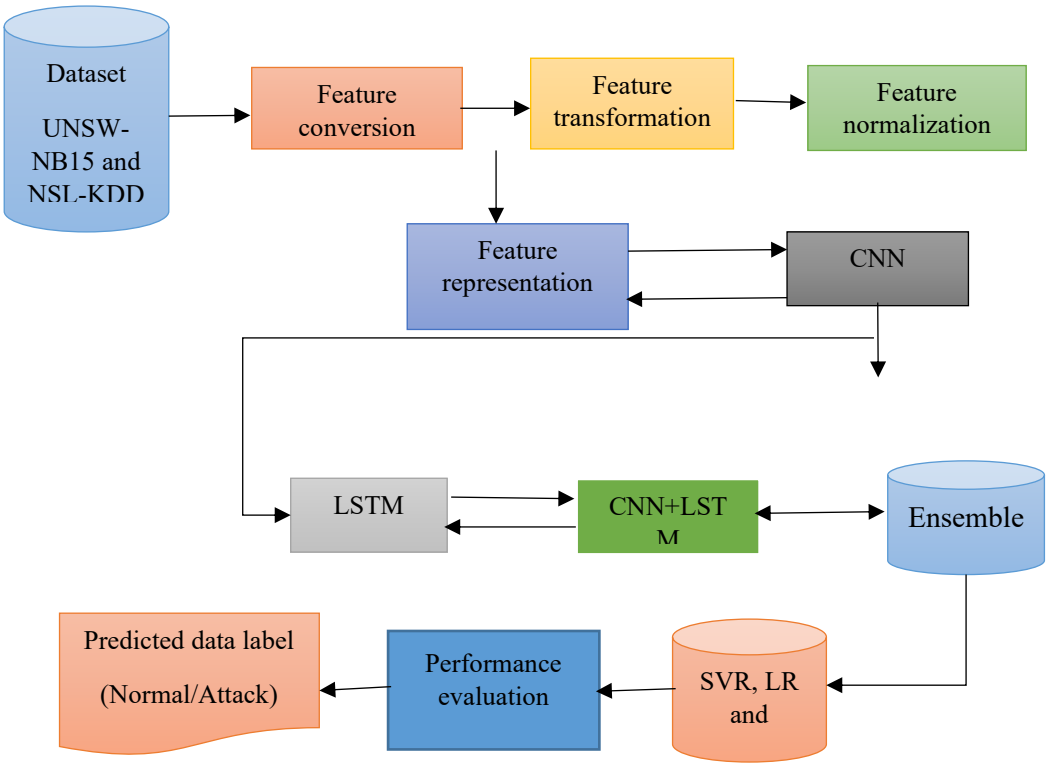

Fig 1: Overview of proposed model

A total of 39 different attack types are assigned to each record. These kinds of attacks fit into any of the following groups, which the authors utilize as the objective variable: DoS, R2L, Probing or observation, and Unauthorized Access to Local Superuser Privileges (U2R), as given.

3.2. Attack prediction using learning model

To address the multi-classification issue, this work provides three DL frameworks in this study that rely on multi-level data fusion processes. They all begin with a standardization layer that converts the decimal attributes into a distribution with a center of 0 and a standard deviation of 1. For the string scores and integer, lookup layers use tables to transform classified data values into their equivalent indices. More specifically, unlike conventional fully connected networks, this work handles the input features independently based on their nature. For instance, the dataset is separated into subgroups according to the kind of data, and a distinct preprocessing layer prepares each subset. On the other hand, raw data is directly merged and utilized as network source data in a classic, fully linked network. By simplifying the local framework of the data, where instances from an identical subset share a similar data type, our proposed preprocessing phase aims to make it easier for the fully connected network to understand the links among the various feature types.

3.3. Traditional Convolutional Neural Networks (CNN)

The CNN is the most effective selective algorithm, which uses both pooling and convolutional layers. On top of one another is the arrangement of the layers. At the same time, the combined layers smooth the convolutional layer discharge and lower the quantity of data rate from the lowermost layer; the convolutional layer distributes much weight. The CNN has some invariance qualities owing to the convolutional layer's weight allocation and well-selected pooling algorithms. One of the best DL neural networks for automated vision and recognition of images has been identified as CNN. In CNN's convolutional layer, learnable kernels are convolved with convolutional characteristics mapping from the preceding layer with rectified linear unit or nonlinear activation functions like softmax, tanh, and sigmoid across various approaches to provide resulting feature maps. Consequently, each feature map output is blended with several feature map inputs. As given in the following Eq. (1):

$$x_j^i = f\left(\sum_{i \in M_j} x_i^{I-1} * k_{ij}^I + b_j^I\right) \quad (1)$$

Where, k_{ij}^I represents the kernel for the current layer, b_j^I is the biases for the present layer, x_j^i is the result of the present layer, and x_i^{I-1} is the result produced by the preceding layer. M_j denotes selection input maps; every resulting map is denoted by an addictive bias b. Since, there has to be specifically n resultant maps if it contains n input maps, the total amount of input and output feature maps at the CNN pooling layer cannot be altered. However,

because of the down-sampling process, the resulting map's length will be smaller. The following equation can be used to describe the operation:

$$x_j^i = \text{down}\left(x_j^{I-1}\right) \quad (2)$$

In addition, CNN always uses convolution, downsampling, and subsampling processes as its activation functions to integrate nonlinearity.

3.4. LSTM

An RNN is an LSTM that has three stages: input (a series input layer that provides data to the network's neural network in the format of a temporal sequence), forget (this stage decides which data is essential and which can be ignored), and output (this stage calculates the outcome of the following hidden state and includes information gathered from previous inputs). The data flow at time step t is shown. These graphs further demonstrate how the gates output cell and hidden states, update and forget. Three stages make up the LSTM.

$$f_t = \sigma\left(w_f\left[h_{t-1,\, x_t}\right] + b_f\right) \quad (3)$$

$$i_t = \sigma\left(w_i\left[h_{t-1,\, x_t}\right] + b_i\right) \quad (4)$$

$$C_i = \tanh\left(w_t\left[h_{t-1,\, x_t}\right] + b_c\right) \quad (5)$$

$$C_t = f_t * C_{t-1} + i_t * C_t \quad (6)$$

$$O_i = \sigma\left(W_o\left[h_{t-1,x_t}\right] + b_o\right) \quad (7)$$

$$h_t = O_t * \tanh*(C_t) \quad (8)$$

First, x(t) and h(t-1) determine which data should be deleted. Eq. (3) verifies this. Second, the newly added data input layer is turned on. The sigmoid function in Eq. (4) is used to reorganize the data. The tanh function yields the candidate data that arises with the recently acquired data. Eq. (5) is then used to create the fresh data. Finally, Eq. (7) identify the outcome. The framework then learns its weight variables (W) and bias variables (b) to reduce the difference between the LSTM outcomes and real-world training scores.

3.5. Hybridization

An influential CNN-LSTM association was suggested to significantly identify and structure the short- and long-term spatial interconnections exhibited in the series of datasets. This combination limits the use of CNN for gaining valuable information and acquiring capability with superior effectiveness of LSTM protocol in a temporal-sequence-internal-representation-dependent method. To accomplish the goal above, the suggested CNN-LSTM comprises the following two essential parts: Convolutional and pooling layers comprise a 1D CNN, which applies a mathematical process to the input information to produce features; Employing LSTM and thick layers to utilize the features that CNN generated is used as the encoder, and LSTM is used as the decoder in the suggested CNN-LSTM algorithm. The input data is used to teach the encoding device a feature, which is then sent into an LSTM decoder. Subsequently, the decoder recognizes and analyzes the dataset's underlying short- and long-term temporal relationships. Below is a quick summary of each stage's series of events.

1) The input layer is where the data is received.
2) First Convolution Layers: Observe the input data from before applying the results to the feature maps;
3) Finding the feature map again, the second convolution layer uses 32 feature maps/convolution layer and a kernel dimension of 3 times to scan the data source pattern to improve

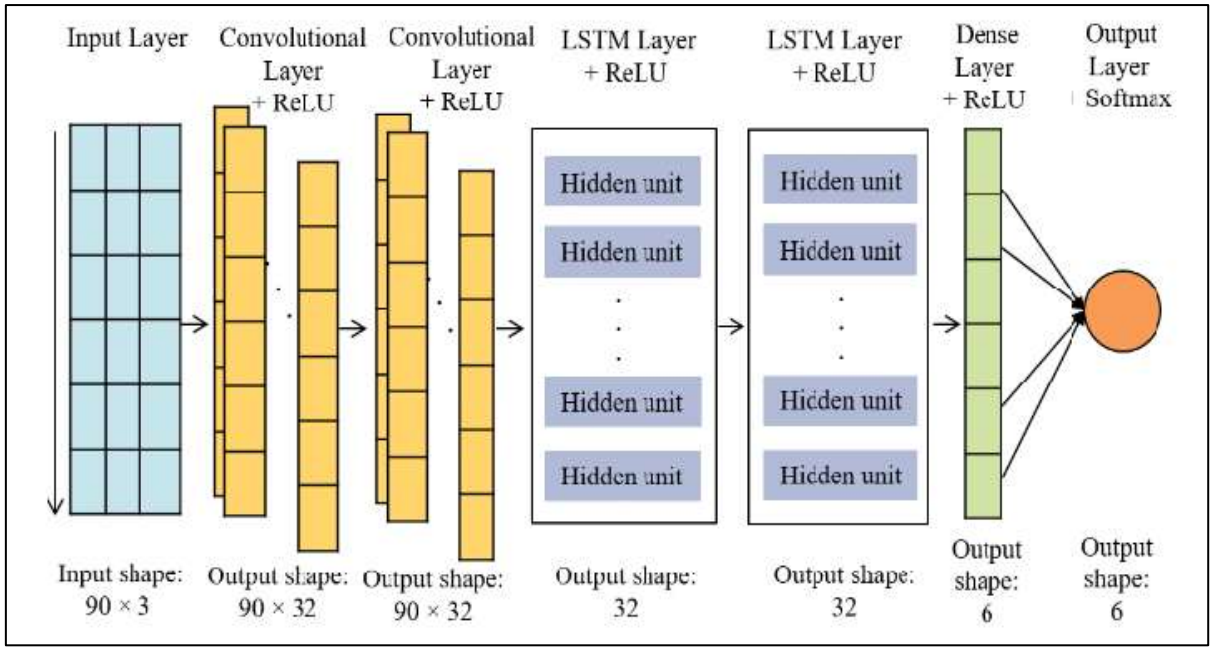

Fig 2: Hybridization of proposed model

4) Max pooling layer: Oversimplifies the feature maps and creates matrices with a short size by eliminating some characteristics from (iii) above;
5) Dropout layer: Improves the learning network to avoid over-fitting the technique;
6) Flatten layer: Generates a single, long vector from the distilled feature maps that can be used as input for decryption;
7) Repeat Vector layer: the interior model of the data input pattern is recurring once for each time step in the resultant series;
8) The LSTM decoding device has one hundred hidden layers that can output the entire structure, each of which has a total of 100 units offering an integer daily, which serves as a foundation for predicting what will occur in the upcoming days in the resultant order;
9) Fully connected layer: understands every step in the series of outputs to create identical layers for identifying a particular sequence result, demonstrating that the LSTM decoding device.
Here, $\propto^*$and$\propto$ represents Lagrange multipliers, A represents training input, Y represents training output, b represents bias parameters, $K(x_i,x)$ represents kernel function and f(x) represents SVR equation.

$$\max_{\propto,\propto^*} -\left(\frac{1}{2}\right)(\propto^*-\propto)^T K(A,A^T)(\propto^*-\propto)+Y^T(\propto^*-\propto)+\in e^T(\alpha^*+\alpha) \quad (12)$$

$$e^T(\propto^*+\propto)=0,\ 0e\leq \propto, \propto^* \leq Ce \quad (13)$$

$$f(x)=\sum_{i=1}^{n}(\propto^*-\propto)K(x_i, x)+b \quad (14)$$

can work at any given moment, similar to both output and FCN layers.
10) Output layer: This layer forecasts the number of cases that occur for new illness and deceased cases over ten days.

3.6. Classification

3.6.1 SVR

In hyperspace classification, the non-linear SVR looks for a regression function given as $f(x)=w^T\phi(x)+b$. The "ϵ-insensitive" loss function computes the equation. The nonlinear SVR is expressed by the equations below.

3.6.2. Linear regression

Along with this, simple linear regression uses a mathematical formula to represent the linear connection between a dependent factor (y) and an independent single factor (x). The link between several independent variables (x_n) and the dependent factor (y) can be expressed using the MLR approach. For independent m variables, the preceding equation represented a mathematical framework illustrating the connection in a multivariate regression study.

$$Y=h_0+h_1x_1+h_2x_2+\ldots+h_mx_m+t \quad (15)$$

Here, Y represents dependent variables, X represents independent variables, h_o represents regression curve, m represents input parameters, t represents error team and $h_m$xm represents regression coefficient of independent variables.

3) Boosting model

To achieve a better prediction result, the gradient-boosting ensemble technique consists of a limited set of meta-learners and weak learners that assigns weights to all learners before combining their estimator results using voting procedures. One technique that employs least squares as its loss criterion is called "least-squares boosting". The investigator was the first to employ the LS - Boost approach. Friedman recommended applying the LS-Boost approach with the following equations. It is necessary to define the number of iterations (H) and the explainable variables (x_i and y_i). The following equations are then used to define the training set, a loss function, and a regression function, respectively.

$$\{(x_i, y_i)\}_{i=1}^{n} \quad (16)$$

$$L(y, F)= \frac{(y-F)^2}{2} \quad (17)$$

$$F_m(x) \quad (18)$$

Initially, set $F_o(x) = \bar{y}$; finally, it is computed with the following equations.

t=1 to H do: $\tilde{y}_i - F_{m-1}(x_i)$ for I=1, 2, …,N (19)

$$(\rho_t, \alpha_t)= \arg\min_{\rho,\alpha} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(x_i;\alpha)]^2 \quad (20)$$

$$F_t(x)=F_{t-1}(x)+ \rho_t h(x; \alpha_t) \quad (21)$$

Here, x_i and y_i represents explainable variables, H represents iteration, $\{(x_i, y_i\}_{i=1}^{n}$ represents training set, L represents loss function, $F_m(x)$ represents regression function, h represents activation function, α represents random sequences and ρ represents learning rate. Sensitivity testing is valuable for determining how experimental variables affect the outcome. The research's sensitivity evaluation employed the cosine amplitude approach. The cosine amplitude approach helps specify how experimental parameters affect the outcome. It uses n data specimens to create a collection of data specimens.

$$X=[x_1, x_2,\ldots,x_n] \quad (22)$$

In the data array X, every element, x_i is a vector of size m.

$$X_i=[x_{i1}, x_{i2},\ldots,x_{in}] \quad (23)$$

The dependent and independent variables in the data are denoted by x_j and x_i, respectively. Therefore, each element's identity is calculated using the following equation In the data array X, every element, x_i is a vector of size m.

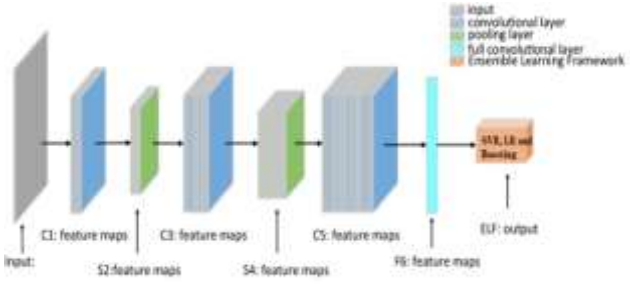

Fig 3: Proposed model architecture

4. RESULTS AND DISCUSSION

This work used the Recall, Accuracy, and Precision measures for the multiclass confusion matrix, where the source data can be categorized into any of the (c) classes to assess the effectiveness of our DL models. Consequently, the following is the definition of the matrices:

$$\text{Accuracy}= \frac{1}{c}\sum_{i=1}^{c} \frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i} \quad (24)$$

$$\text{Precision}= \frac{1}{c}\sum_{i=1}^{c} \frac{TP_i}{TP_i+FP_i} \quad (25)$$

$$\text{Recall}= \frac{1}{c}\sum_{i=1}^{c}\frac{TP_i}{TP_i+FN_i} \quad (26)$$

Where, TP and TN indicate accurate predictions, FP and FN indicate inaccurate predictions of the model, and c is the total classes present in the dataset (i.e., ten distinct classes). Based on a particular class TP, FP, TN, and FN are defined as follows:

$$TP_i= \sum_{i=1}^{c} x_{ii} \quad (27)$$

$$TN_i=\sum_{i=1}^{c}\sum_{k=1}^{c} x_{jk}, \quad j\neq i, \ k\neq i \quad (28)$$

$$FP_i= \sum_{j=1}^{c} x_{ji}, \quad j\neq i \quad (29)$$

$$FN_i= \sum_{j=1}^{c} x_{ji}, \quad j\neq i \quad (30)$$

Where, x_{ii} denotes an element located diagonally on the multi-classes confusion matrices. Lastly, for an accurate comparison, a comparable hyperparameter adjustment technique produced the total amount of neurons in every single layer even if our models' deep learning topologies differed. This study used the MATLAB 2020a module to examine the optimal hyper-parameters for deep neural network models. Additionally, all models were trained with a batch size ranging from 32 to 100 epochs using the L2 regularization approach (at 0.01). This section examines and evaluates the accuracy of the suggested models for DL shows how well our data fusion techniques work and how reliable it is to combine several models into one deep learning model and finally contrasts our models' achievement with that of the most advanced models. The DL approach of conventional fusion achieved 99.9 percent precision in the training dataset, 84.3 percent in the validation dataset and 76.3 percent in the test set as shown in Table 1. Nonetheless, the late fusion approach achieved a precision of 85.9% during training, 83.9% during validation and 77.1% during testing. The difference between the precision or failure patterns on the verification and validation is considerably smaller in the proposed late-fusion approach compared to the early-fusion model as Fig 4 to Fig 6 further illustrates. This demonstrates the late-fusion framework's capacity to choose more general traits. Additionally, it performs better on the verification dataset. The accuracy of the proposed integrated approach was 76.7 percent on the testing set, 84.6 percent on the training dataset and 83 percent on the validation dataset. Furthermore, this work experimented with a different dataset (NSL-KDD) to further test our three suggested approaches- early-fusion, late-ensemble and late-fusion used the MATLAB 2020a module to choose the best collection of hyper-parameters. The findings are shown in Table 1, show that the early-fusion neural network model worked better on the testing dataset than the late-fusion approach. Conversely, the late ensemble model performed best with recall, accuracy, and precision rates of 86%, 87%, and 88%, respectively. Additionally, this work used the late-ensemble approach to construct the ROC curves that illustrate the sensitivity to the individual classes.

4.1. Comparison to existing models

After that, this work compares our DL models with existing models are ensuring they are fair by using models already suggested for the multi-classification mission of differentiating between different attacks in the UNSW-NB15 and NSL-KDD dataset testing sets. The Adaboost approach performs the worst on the UNSWNB15 test set, whereas the DT ensemble C-LSTM attains the maximum accuracy of 84.32% as per the data shown in Table 1. Nevertheless, recall and precision also show very low scores suggesting a restricted capacity to deal with the problem of class bias. On the other hand, our suggested model achieves better recall and precision scores of 69.50% and 86.04%, respectively demonstrating its ability to manage the problematic dataset without the need for preprocessing. Concerning the NSL-KDD dataset, the performance of the various models under comparison is similar; the AE approach achieves the best precision of 87.85%, while the QDA approach yields the lowest accuracy of 64.37%. Nevertheless, our model outperforms all previous approaches regarding recall (86.80%) and accuracy (86.81%).

4.2. Discussion

To increase a NIDS's effectiveness in identifying and characterizing malicious behavior, this work addresses practical aspects in this section while utilizing the DL-based technique previously mentioned. As previously stated, the present NIDS integrates anomaly-, signature-, and protocol analysis-based detection techniques. On the other hand, the framework has suggested is a multi-way classifier where one class symbolizes typical behavior and the others reflect various forms of malevolent behavior. In a way, multi-way classifiers fall somewhere between anomaly-based and signature-based methods. Although, it could be more precise and fine-grained than signature-based approaches, it utilizes knowledge of previous attacks to facilitate data that helps identify the type of threat. It is behavior-based and employs DL, just like anomaly-based techniques. However, it allocates examples to a specific set of classes of which one is regular and the remaining are malevolent, instead of recognizing normal and

classifying everything else as "other." The technique should be far less susceptible to minute modifications in the threat process than signature-based methods, making it more complex for an intruder to escape. When confronted with a new assault that does not fit into the categories in the training dataset, however, the "closed world" statement may be problematic.

Table 1: Performance with UNSW-NB15 dataset

DL model	Train	Validation performance (%)	Test
CNN			
Accuracy	91	85	77
Precision	96	91	84
Recall	88	81	72
LSTM			
Accuracy	86	84	78
Precision	94	92	86
Recall	82	80	70
Ensemble C-LSTM			
Accuracy	94	94	96
Precision	92	91	86
Recall	89	98	98

Table 2: Performance with NSL-KDD dataset

DL model	Train	Validation performance (%)	Test
CNN			
Accuracy	92	86	78
Precision	97	92	85
Recall	89	82	73
LSTM			
Accuracy	87	85	79
Precision	95	93	87
Recall	83	81	72
Ensemble C-LSTM			
Accuracy	95	95	97
Precision	93	92	87
Recall	90	99	98

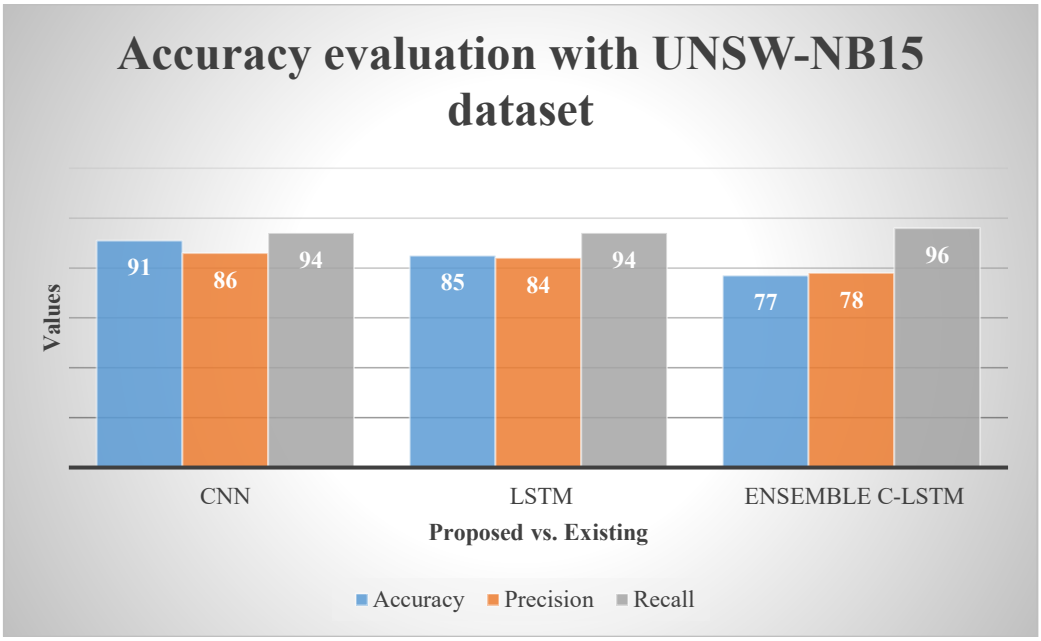

Fig4: Accuracy evaluation with UNSW-NB15 dataset

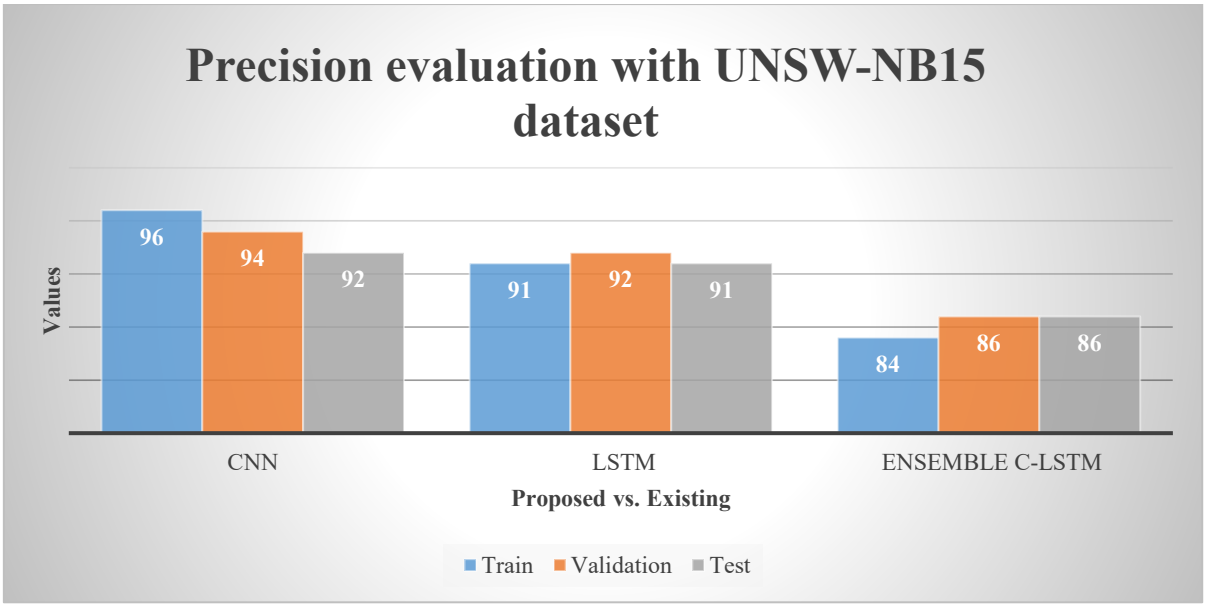

Fig 5: Precision evaluation with UNSW-NB15 dataset

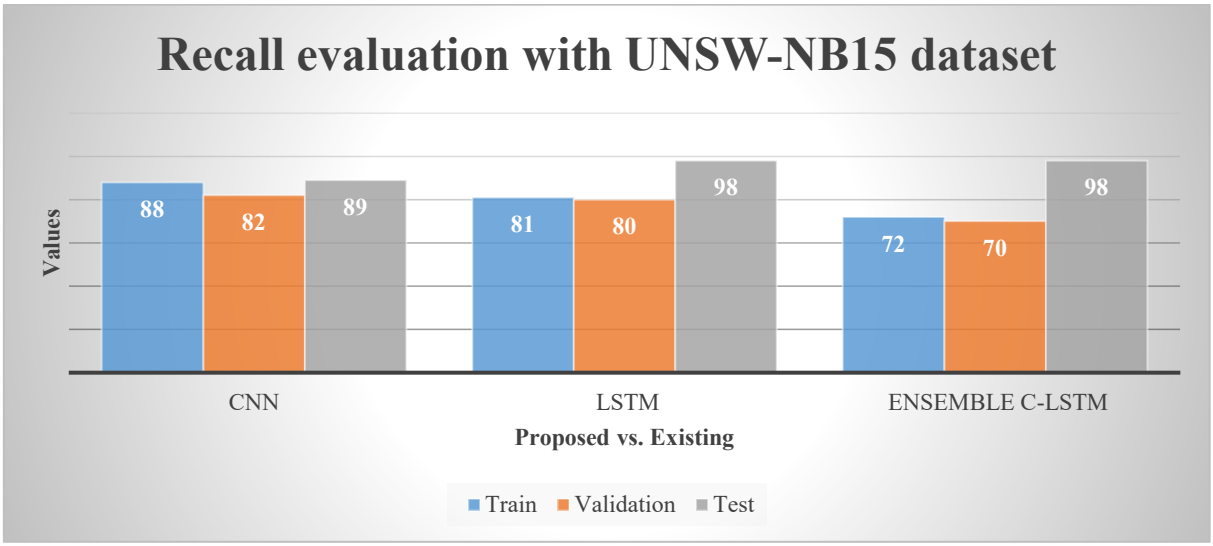

Fig 6: Recall evaluation with UNSW-NB15 dataset

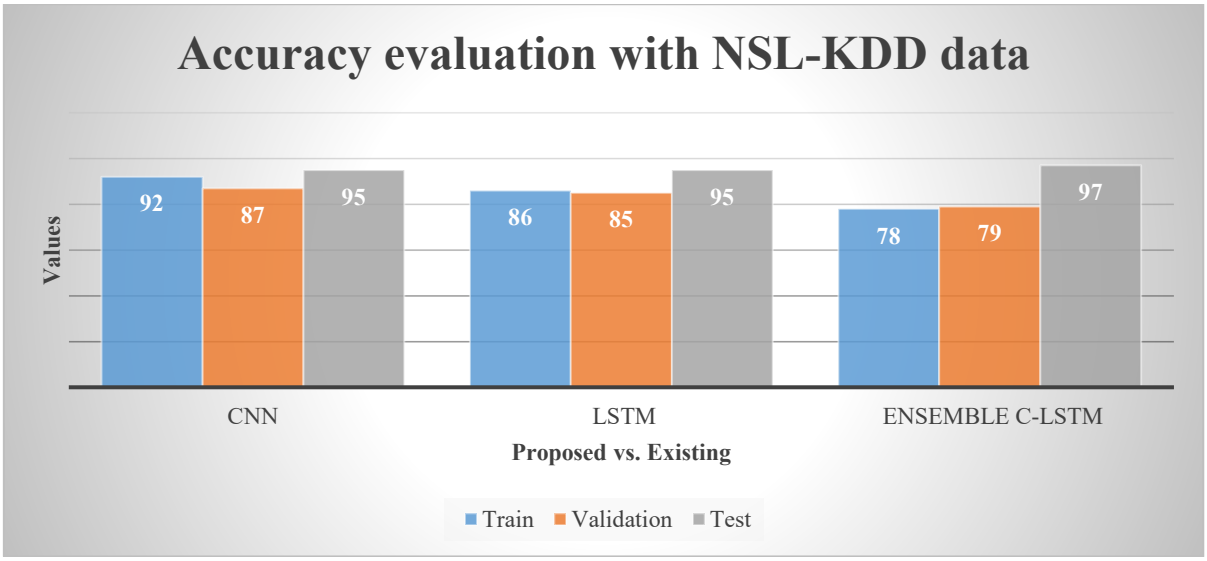

Fig 7: Accuracy evaluation with NSL-KDD dataset

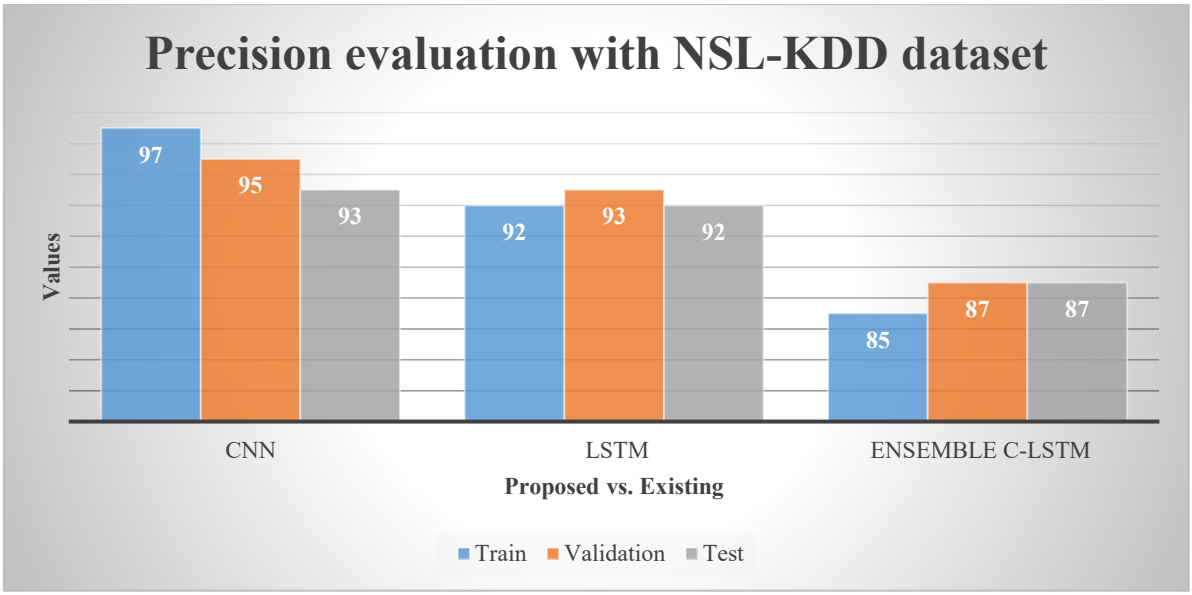

Fig 8: Precision evaluation with NSL-KDD dataset

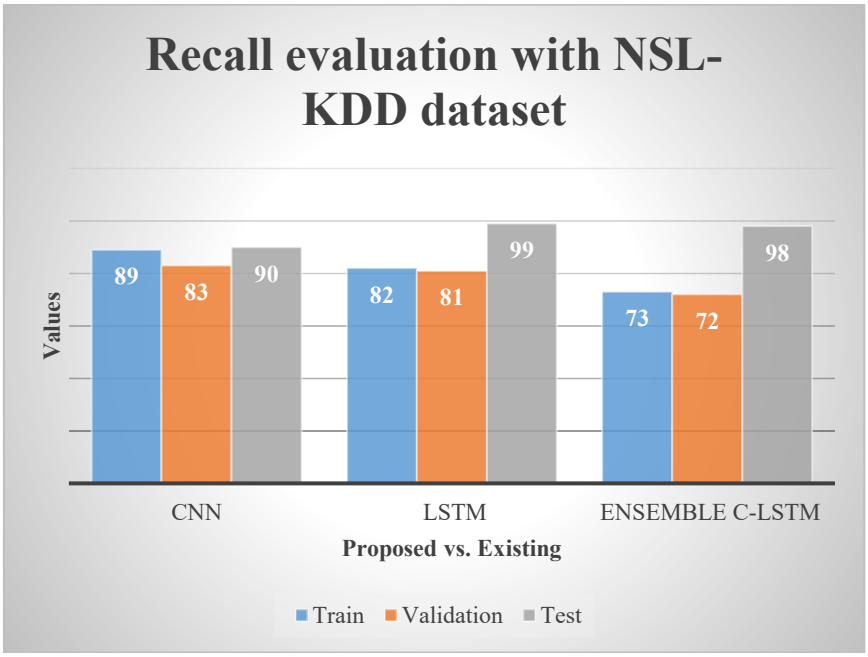

Fig 9: Recall evaluation with NSL-KDD dataset

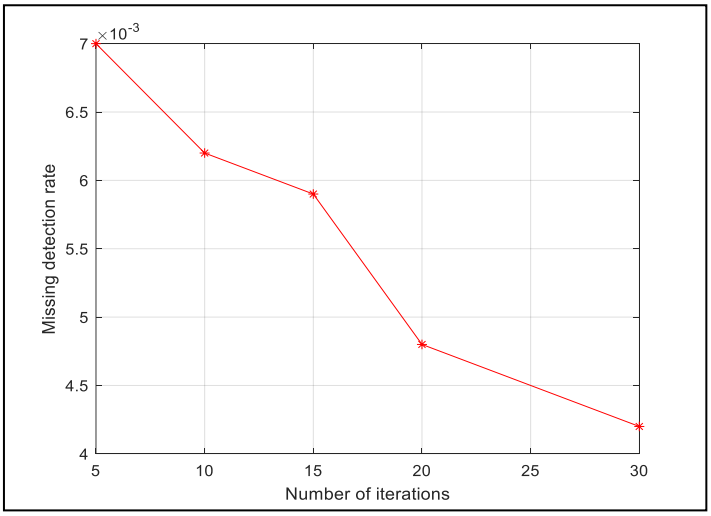

Fig 10: Detection rate

Anomaly-based methods need ongoing modification to reduce false positives and may require a long training period to adjust to a different deployment setting. Given the constantly shifting threat environment, it appears likely that regular and threat behavior profiles that a multi-way classifier learns are context-sensitive. Thus, it is a problem to generate or capture and label training data for the assault kinds. More research is needed to determine the extent of generalization. For example, it is necessary to determine whether the classification algorithm is simply learning threat signatures, is effectively generalizing to find more abstract features of the different attack categories utilized to generate a training dataset at a deeper level is identifying some intrinsic distinction between fraudulent and regular congestion.

5. CONCLUSION

Using deep fusion techniques known as early-fusion ensemble deep learning models, this study suggests new deep learning structures. Our feature fusion algorithms recommend that the DL model captures the links among the features. As they can learn the associations between more specific features, the late-ensemble and late-fusion approaches outperformed the early-fusion models and other modern models. Our feature-fusion-based deep learning structures offer a general solution that can be applied to Ensemble C-LSTM among other DL models. The effect of feature fusion employing recurrent modules to encode the lengthy relationships among the features will be investigated as a future development. By emphasizing the most important characteristics that contribute to the final prediction, post-hoc explainable AI models can be used to understand better how the suggested models behave when differentiating across classes

Declarations

Ethical Approval

The dataset samples used in this study are publicly available from the UNSW-NB15 and NSL-KDD repositories. Therefore, this research did not require ethical approval.

Competing Interests

The authors declare that they have no competing interests.

Conflict of Interest

The authors declare that they have no conflict of interest.

Authors' Contributions

Arun Prasad: Conducted the literature survey, designed and implemented the proposed framework, and prepared the initial draft.

Pattabirani: Assisted in methodology development, coding, and experimental validation.

Sundaramoorthy: Provided supervision, critical review, and finalized the manuscript for submission.

Funding

None

Availability of Data and Materials

The datasets analyzed during the current study are publicly available at:
UNSW-NB15: https://research.unsw.edu.au/projects/unsw-nb15-dataset
NSL-KDD: https://www.unb.ca/cic/datasets/nsl.html

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.
[2] Y. Ramadevi Reddy and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICAC), Sep. 2016, pp. 1148–1153.
[3] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in Proc. Int. Conf. Signal Process. Commun. Eng. Syst., Jan. 2015, pp. 92–96.
[4] F. Farnaaz and M. A. Jabbar, "Random forest modelling for network intrusion detection system," Procedia Comput. Sci., vol. 89, pp. 213–217, Jan. 2016.
[5] A. Khan and N. Jain, "A survey on intrusion detection systems and classification techniques," Int. J. Sci. Res. Sci. Eng. Technol., vol. 2, no. 5, pp. 202–208, 2016.

[6] T. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for network intrusion detection in software-defined networking,” in Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM), Oct. 2016, pp. 258–263.
[7] A. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, “Fuzziness based semi-supervised learning approach for an intrusion detection system,” Inf. Sci., vol. 378, pp. 484–497, Feb. 2017.
[8] A. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, “Fuzziness based semi-supervised learning approach for an intrusion detection system,” Inf. Sci., vol. 378, pp. 484–497, Feb. 2017.
[9] H. Chang, W. Li, and Z. Yang, “Network intrusion detection based on random forest and support vector machine,” in Proc. IEEE Int. Conf. Comput. Sci. Eng./Embedded Ubiquitous Comput., Jul. 2017, pp. 635–638.
[10] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring: A survey,” IEEE Trans. Neural Netw. Learn. Syst. (submitted), 2016. [Online]. Available: http://arxiv.org/abs/1612.07640
[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” J. Mach. Learn. Res., vol. 11, pp. 3371–3408, 2010.
[12] M. Mohan, V. Tamizhazhagan, and S. Balaji, “A perspicacious multi-level defense system against DDoS attacks in cloud using information metric & game theoretical approach,” J. Netw. Syst. Manage., vol. 31, no. 85, 2023. doi: 10.1007/s10922-023-09776-7.
[13] A. Alrawashdeh and C. Purdy, “Toward an online anomaly intrusion detection system based on deep learning,” in Proc. 15th IEEE Int. Conf. Mach. Learn. Appl., Anaheim, CA, USA, Dec. 2016, pp. 195–200.
[14] P. Potluri and C. Diedrich, “Accelerated deep neural networks for an enhanced intrusion detection system,” in Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom., Berlin, Germany, Sep. 2016, pp. 1–8.
[15] T. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for network intrusion detection in software-defined networking,” in Proc. Int. Conf. Wireless Netw. Mobile Commun., Oct. 2016, pp. 258–263.
[16] E. Hodo, X. J. A. Bellekens, A. Hamilton, C. Tachtatzis, and R. C. Atkinson, “Shallow and deep networks intrusion detection system: A taxonomy and survey,” ACM Comput. Surv. (submitted), 2017. [Online]. Available: http://arxiv.org/abs/1701.02145
[17] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon, “LSTM-based system-call language modelling and robust ensemble method for designing host-based intrusion detection systems,” arXiv preprint arXiv:1611.01726, 2016.
[18] 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, 2016.
[19] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, “A deep learning approach for network intrusion detection system,” in Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol. (BIONETICS), New York, USA, vol. 35, 2015, pp. 2126.
[20] S. Otoum, B. Kantarci, and H. T. Mouftah, “Adaptively supervised and intrusion-aware data aggregation for wireless sensor clusters in critical infrastructures,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2018, pp. 1–6.
[21] A. Gouveia and M. Correia, “A systematic approach for the application of restricted Boltzmann machines in network intrusion detection,” Lecture Notes in Computer Science, vol. 10305, May 2017.
[22] M. Beigh and M. A. Peer, “Performance evaluation of different intrusion detection system: An empirical approach,” in Proc. Int. Conf. Comput. Commun. Informatics, Jan. 2014, pp. 1–7.
[23] W. Zhou, J. Wen, Y. Koh, Q. Xiong, M. Gao, G. Dobbie, and S. Alam, “Shilling attacks detection in recommender systems based on target item analysis,” PLoS One, vol. 10, no. 7, p. e0130968, Jul. 2015.
[24] T. Kumari and P. Bedi, “A comprehensive study of shilling attacks in recommender systems,” Int. J. Comput. Sci. Issues (IJCSI), vol. 14, no. 4, pp. 44–50, 2017.
[25] J. Cao, Z. Wu, B. Mao, and Y. Zhang, “Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system,” World Wide Web J., vol. 16, no. 5–6, pp. 729–748, 2013.
[26] H. Yu, R. Gao, K. Wang, and F. Zhang, “A novel robust recommendation method based on kernel matrix factorization,” J. Intell. Fuzzy Syst., vol. 32, no. 3, pp. 2101–2109, 2017.
[27] Z. Yang and Z. Cai, “Detecting abnormal profiles in collaborative filtering recommender systems,” J. Intell. Inf. Syst., vol. 48, no. 3, pp. 499–518, 2017.
[28] W. Zhou, J. Wen, Q. Qu, J. Zeng, and T. Cheng, “Shilling attack detection for recommender systems based on credibility of group users and rating time series,” PLoS One, vol. 13, no. 5, p. e0196533, May 2018.
[29] A. Turk and A. Bilge, “Robustness analysis of multi-criteria collaborative filtering algorithms against shilling attacks,” Expert Syst. Appl., vol. 115, pp. 386–402, 2019.
[30] P. Moradi and S. Ahmadian, “A reliability-based recommendation method to improve trust-aware recommender systems,” Expert Syst. Appl., vol. 42, pp. 7386–7390, 2015.