International Journal of Advanced Trends in Engineering and Management

(ISSN 2583-7052)

Vol. V, No. 03, pp. 9-19, March

Deep Learning-Based ResNet- BiLSTM Framework for Network Intrusion Detection

R. K. Padmashini[3]

[3]Assistant Professor, Department of Electrical and Electronics Engineering,
AMET Deemed to be university, Chennai, Tamil Nadu, India - 603112.

padmashinirkeee@gmail.com

ABSTRACT: Intrusion detection is a security method that monitors and analyzes system or network activity to identify malicious attacks, unauthorized access, or policy violations. It is essential for safeguarding digital infrastructures against continuously evolving cyberthreats. Traditional Intrusion Detection Systems (IDS) often struggle with poor scalability under heavy network traffic. They also suffer from high false alarm rates and weak adaptability to new attack patterns. These drawbacks make them less reliable in complex environments and limit their real-time detection capability. To tackle these issues, this paper proposes an optimized Residual Network – Bidirectional Long Short-Term Memory (ResNet-BiLSTM) model. The study is conducted using the benchmark NSL-KDD dataset for evaluation. Preprocessing steps include addressing missing values, applying Min-Max normalization, and performing one-hot encoding. A balanced dataset is then generated to minimize the adverse effects of class imbalance. The Fennec Fox Optimization Algorithm (FFOA) is applied for effective feature selection, ensuring the most relevant inputs are retained. The optimized ResNet-BiLSTM captures both spatial and temporal dependencies in the processed features. The proposed framework achieves precision of 0.96, recall of 0.94, F1-score of 0.95, and accuracy of 0.94. Implemented in Python, the system reduces false alarms and delivers robust performance for modern intrusion detection.

Keywords: Intrusion Detection Systems (IDS), Residual Network – Bidirectional Long Short-Term Memory (ResNet-BiLSTM), Fennec Fox Optimization Algorithm (FFOA)

1. Introduction

Identifying unwanted use or attacks on a computer or telecommunications network is known as intrusion detection systems (IDS). IDSs are software or hardware solutions made to reduce these risks by identifying attempts to jeopardize the availability, confidentiality, or integrity of information resources [1]. An essential component of everyday life and organizational operations is the wide spread use of networked computer systems. At the same time, it raised questions regarding user's internet security and privacy. According to current polls, there were roughly 5.1 billion cyberattacks in 2021. Additionally, the data show an increase in highly effective and sophisticated cyberattacks on vital infrastructure around the world [2]. IDS are generally divided into two types: host-based IDS (HIDS) and network-based IDS (NIDS), are essential in combating these expanding threats. HIDS monitors activities and changes on individual devices to detect malicious behavior, whereas NIDS observes network traffic to identify suspicious patterns and potential intrusions [3]. Security teams are utilize additional IDSs for

particular duties in addition to the usual NIDS and HIDS. One such option is a Protocol-based IDS (PIDS), which monitors the connection protocols between servers and devices. PIDS is typically installed on web servers to keep an eye on HTTP or HTTPS traffic. An alternative is the Application Protocol-based IDS (APIDS), which operates at the application layer and analyzes protocols specific to a particular application. An APIDS is typically deployed between a web server and a SQL database in order to detect SQL injections [4]. Building upon these traditional IDS categories, earlier detection mechanisms often relied on classical machine learning approaches. Algorithms such as K-Nearest Neighbors (KNN), Random Forest etc., have been widely applied to classify normal and malicious network activities. These techniques utilize predefined patterns and feature extraction to detect intrusions [5]. The reliance of traditional machine learning algorithms on manually created features limits their capacity to adjust to hidden attack patterns. These models frequently have trouble identifying new or zero-day threats outside of their training data as cyberattacks continue to change. Additionally, when subjected to complicated or loud network data, they generate a high proportion of false positives. Their total efficacy thus declines in dynamic settings that necessitate ongoing learning and flexibility [6]. Intrusion detection systems are increasingly using deep learning techniques to overcome these issues. In contrast to conventional techniques, they can automatically extract intricate temporal and geographical properties from unprocessed network data. This enables them to provide enhanced resistance, adaptability, and detection accuracy against complex and ever-evolving cyberthreats [7-8]. Moreover, recent research emphasizes the importance of scalability and real-time adaptability in intrusion detection systems to effectively handle the continuously growing volume of network traffic [8]. Studies also highlight the role of IDS in securing emerging domains such as IoT and cloud computing, where evolving attack vectors pose significant risks [9-10]. In addition, integrating IDS with intelligent threat intelligence platforms has been shown to improve resilience, reduce false alarms, and enhance the overall trustworthiness of networked infrastructures [11-12]. In order to overcome these constraints, the suggested system presents an Optimized ResNet-BiLSTM architecture for NIDS on the benchmark NSL-KDD dataset.

The key contributions of this proposed work are summarized below, highlighting its innovative aspects and the improvements it offers over existing methods.

- To use the NSL-KDD benchmark dataset to create an ideal network intrusion detection system.
- To handle missing values, normalize features, and encode categorical variables in order to analyze and generate high-quality incoming data.
- To perform effective feature selection using the Fennec Fox Optimization Algorithm (FFOA) to enhance feature relevance.

2. Literature Survey

Asmaa Halbouni *et al* [2022] [13] have presented an IDS based on the Convolutional Neural Network (CNN) and LSTM deep learning algorithms. In this model, combined stacked CNN and LSTM layers, taking advantage of the CNN's prowess in capturing spatial characteristics and the LSTM's capacity to identify patterns across time. This integration improves the model's capacity to recognize intricate attack behaviors by enabling it to efficiently learn both short-term and long-term patterns in network traffic data. In contrast to simpler structures, the method has drawbacks including longer training times and higher processing requirements. Additionally, handling extremely unbalanced datasets or noisy input data may have an impact on its performance.

Vanlalruata Hnamte *et al* [2023] [14] have developed a two-stage intrusion detection system that analyze network activity and makes use of a very effective framework. For real-time data processing and analysis, the system employs a distributed deep learning framework that incorporates CNN and DNN models for a thorough comparison with the suggested LSTM-AE model.. This architecture enhances the system's capacity to discriminate between benign and malevolent activity, which qualifies it for intricate incursion situations. Nevertheless, in contrast to simpler models, the method requires more processing power and longer training periods. Additionally, extremely dynamic or quickly changing attack patterns may pose a threat to its performance.

Zhen Wang *et al* [2023] [15] have established an attention-based CNN intrusion detection model. An accurate and efficient processing flow is created when combined with the picture generating techniques covered in this study. By concentrating on the most pertinent patterns in network traffic, the attention-based CNN intrusion detection model enhances feature representation and increases detection resilience and accuracy. Its capacity to rank important features in order of importance aids in spotting subtle assault indicators that traditional CNN models could miss. Furthermore, if the attention weights are not successfully learnt, its performance may deteriorate and be affected by parameter adjustment.

Rachid Ben Said *et al* [2023] [16] have suggested a hybrid CNN and BiLSTM network to enhance NIDS using binary and multiclass classification. The most popular datasets were utilized to test and evaluate the suggested model's efficacy. The results show how well the suggested model works to achieve high accuracy while using little training time. Because of its intricacy, the CNN-BiLSTM model requires a lot of processing power and requires more time to train. If the data is extremely unbalanced or the characteristics chosen are not ideal, its performance also suffer.

Shiming Li *et al* [2023] [17] have proposed a novel CRSF is an anomaly-based IDS framework. A dimension transformation function is used to convert input data into two-dimensional images during the framework's pre-training phase. Convolutional kernels then process these images to extract spatial features. An RNN is then used to extract more detailed temporal information from these feature sequences. The CRSF framework leverages pretrained CNN2D-RNN and SVM to effectively capture spatial-temporal features, achieving high accuracy in Industrial IoT intrusion detection. Its intricate design, however, raises computational overhead and might not be appropriate for resource-constrained real-time applications.

3. Proposed System Modelling and Description

Block Diagram

A proposed Network Intrusion Detection System (NIDS) framework using an Optimized ResNet-BiLSTM model and the NSL-KDD dataset is represented in Figure 1.

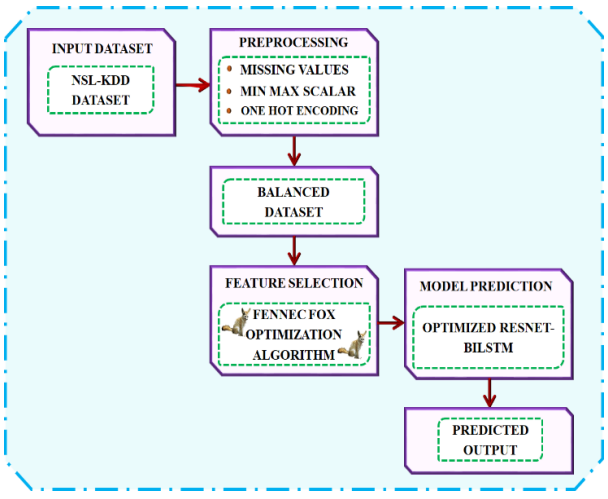

Figure 1 *Block Diagram of IDS using the framework of RESNET-BILSTM*

NSL-KDD dataset is chosen for the input dataset because of its large number of labeled network traffic records. During the preprocessing phase, one-hot encoding is used to encode categorical attributes, Min-Max scaling is used to normalize

numerical features, and missing values are handled. To solve class imbalance issues, the data is subsequently converted into a balanced dataset, guaranteeing that every attack category is equally represented. The Fennec Fox Optimization Algorithm, which effectively finds the most pertinent characteristics while minimizing redundancy and computing load, is used for feature selection. An Optimized ResNet-BiLSTM architecture is employed in the model prediction stage after the feature set has been refined. BiLSTM captures temporal correlations in network traffic patterns, whereas ResNet efficiently recovers deep feature representations. Improved accuracy and resilience in intrusion detection are guaranteed by model optimization. The final phase generates the expected result, categorizing network traffic as either typical or subject to certain kinds of attacks. This method improves overall network security, lowers false positives, and increases detection accuracy.

3.1 Data Preprocessing

Data preprocessing is the process of preparing raw data for accurate and efficient model training. It guarantees that the dataset is structured, consistent, and clean so that analytical algorithms can use it. Managing missing values, which might arise from data entry mistakes, incomplete records, or transmission problems, is a crucial preprocessing activity. In order to keep big values from controlling the learning process, normalization techniques like as the Min-Max Scaler are used to bring numerical characteristics into a predefined range, usually \[0, 1]. Categorical variables are transformed into a binary matrix format using feature encoding techniques like One-Hot Encoding so that algorithms can efficiently analyze them.

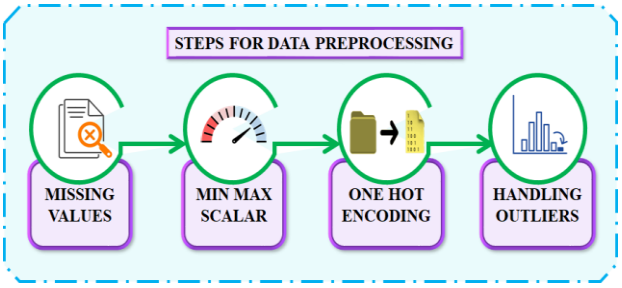

Figure 2 *Steps for data preprocessing*

To mitigate outliers' detrimental effects on model performance, they are detected and eliminated. To enhance feature distribution, data transformation techniques including scaling, normalization, and log transformation can be applied. For objective assessment, preprocessing also entails dividing the dataset into subgroups for testing, validation, and training. If necessary, dimensionality reduction techniques aid in the dataset's simplification while preserving key patterns. This procedure lowers the danger of overfitting and increases computing efficiency. Algorithms can learn efficiently without being mislead by inconsistent or irrelevant information when data is properly preprocessed. All things considered, data preparation serves as the cornerstone of every analytical and has a direct impact on the final model's accuracy and dependability. Even the most sophisticated algorithms might not produce useful results without it. Following preprocessing, the feature selection step is applied to the cleaned and organized dataset. The most important variables influencing model performance are identified in this step. It decreases training time and increases accuracy by eliminating superfluous features.

3.2 Feature Selection-Fennec Fox Optimization Algorithm

After preprocessing, the preprocessed dataset is submitted to feature selection using the Fennec Fox Optimization Algorithm. By emulating the adaptive foraging techniques of fennec foxes, this nature-inspired metaheuristic determines the most significant traits. Fennec foxes comprise the search members of the population-based metaheuristic algorithm known as FFA. Each

fennec fox in FFA stands for a potential solution to the issue, and the values of the decision variables are determined by the fennec fox's location in the search area. Thus, any fennec fox's population is represented by a matrix known as the population matrix, and its mathematical expression is a vector.

$$X_i: x_{i,j} = Ib_j + r.\left(ub_j - lb_j\right), i = 1,2,\dots,N, j = 1,2,\dots,m \quad (1)$$

Where X_i represents the *i*th fennec fox, $x_{i,j}$ denotes its *j*th dimension, *N* is the total number of fennec foxes, *m* is the quantity of factors that are used to make decisions, *r* is an arbitrary number. In (2), a population matrix for the FFA, consisting of *N* fennec foxes, is specified.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1}\dots x_{1,j}\dots x_{1,m} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ x_{i,1}\cdots x_{i,j}\cdots x_{i,m} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ x_{N,1}\cdots x_{N,j}\cdots x_{N,m} \end{bmatrix}_{N\times m} \quad (2)$$

Where $X_i = (x_{i1}, x_{i2}, \dots, x_{im})$denotes the *i*th fennec fox in the population of the range *N* and each column $(x_{ij}, x_{2j}, \dots x_{Nj})^T$ denotes the candidate values for the *j*th decision variable. Consequently, a vector given in (3) is used to mathematically model the estimated values in the objective function.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_t) \\ \vdots \\ F(X_t) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \quad (3)$$

Where F is the objective function's array of variables and F_i is the obtained value of the goal function for the *i*th fennec fox.

$$x_{i,j}^{P1} = x_{i,j} + (2.r - 1).R_{i,j}, \quad (4)$$

$$R_{i.j} = \alpha.\left(1 - \frac{t}{T}\right).x_{i.j} \quad (5)$$

$$X_i = \begin{Bmatrix} X_i^{P1}, F_i^{P1}; \\ X_i, else, \end{Bmatrix} \quad (6)$$

Where X_i^{P1} based on the first phase, is the new recommended status of the ith fennec fox, $X_{i,j}^{P1}$ is the its *j*th dimension, F_i^{P1} is the value of the target function, $R_{i.j}$ is the radius of the neighborhood for $x_{i.j}$, t is the iteration counter, T is the total amount of repetitions, and α is a constant set to 0.2.

$$x_i^{rand}: x_{i,j}^{rand} = x_{k,j}, k \in \{1,2,\dots,N\}, i = 1,2,\dots,N, \quad (7)$$

$$x_{i,j}^{P1} = \begin{cases} x_{i,j} + r.\left(x_{i,j}^{rand} - 1.x_{i,j}\right), & F_i^{rand} < F_i; \\ x_{i,j} + r.\left(x_{i.j} - x_{i,j}^{rand}\right), & else \end{cases} \quad (8)$$

$$X_i = \begin{cases} X_i^{P2}, & F_i^{P2} < F_i; \\ X_i, & else, \end{cases} \quad (9)$$

Where x_i^{rand} is the desired position intended for the escape of the *i*th fennec fox, $x_{i,j}^{rand}$ is the its *j*th dimension, F_i^{rand} is the target's worth, x_i^{P2} is the newly recommended status of the *i*th fennec fox based on second phase, $x_{i,j}^{P2}$ is its *j*th dimension, F_i^{P2} is the value of its objective function, and *I* is a random number from the set {1,2}.

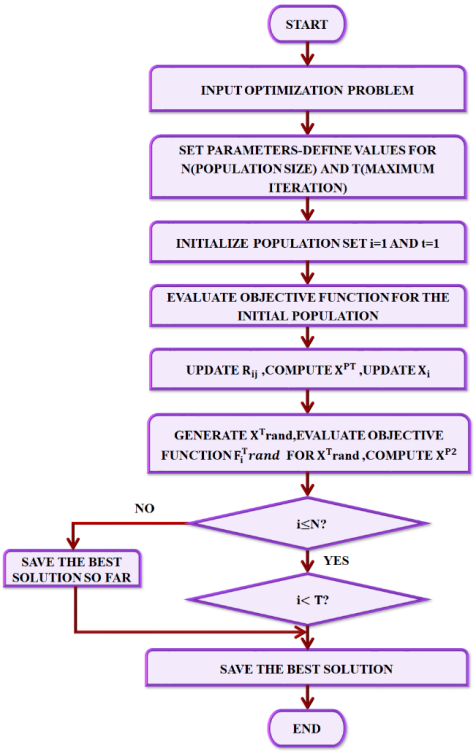

Figure 3 *Flow chart for Fennec foxes Optimization Algorithm*

3.3 Model Prediction Optimized ResNet BiLSTM

The improved dataset is sent to the model prediction step once the feature selection procedure is finished. For the ResNet-BiLSTM model, the initial input is the image stream formed by the conversion of multiple traffic streams, and the final output is the classification results of these traffic streams. The three primary modules of the ResNet-BiLSTM model put out in this work are the ResNet module, the BiLSTM module, and the attention mechanism module. These modules are built in the following order: ResNet module-attention mechanism module-BiLSTM module, and the overall structure of the model.

3.3.1 ResNet

ResNet is a DL model. It effectively tackles the problem of deterioration with increasing network depth and demonstrates strong classification capabilities. The ResNet network, which creates feature maps using temporal sequence information by carrying out convolution operations in the time dimension. Corporates' batch normalization (BN) layers use the ResNet model to create correlations between batch samples. Therefore, the output of the model is not uniquely determined for a particular training sample during random batch processing in the network. This feature aids in preventing overfitting problems to a certain level.

$$y = f(x, \mathrm{W}) + x$$

$$H(x) = g(y) \qquad (10)$$

Where x indicates the prior module's data input; W is the weight matrix of the convolutional layer; f(x, W) represents the result of convolutions of x; y denotes the remaining output; g refers to activate the function of ReLU; $H(x)$ denotes the result of running x through the ResNet model.

3.3.2 BiLSTM

BiLSTM is derived from LSTM, a recurrent neural network (RNN) type. Resolving the issues of exploding and vanishing gradients that frequently arise in RNNs when working with lengthy sequence data is the design objective of LSTM. "Forget gate," "input gate," and "output gate" are among the gate control methods introduced by LSTM that allow long-term memory to process lengthy sequences of data.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \qquad (12)$$

Finally, the memory cell, C_t, takes place using the $tanh$ activation mechanism .

$$h_t = o_t \times tanh(C_t) \qquad (13)$$

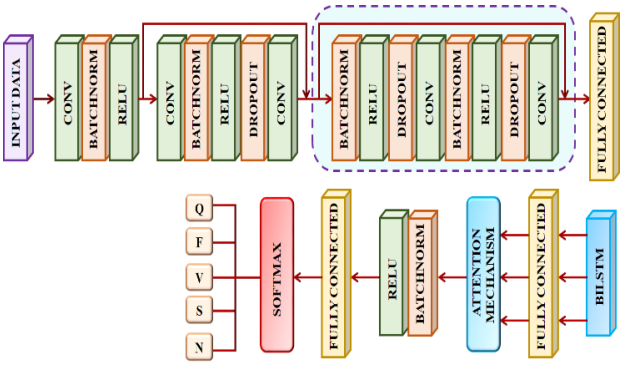

Figure 4 *Structure of ResNet-BiLSTM model*

4. Results and Discussion

According to the Findings, the proposed solution performs better than traditional methods. The results show increases in precision, effectiveness, and general dependability. These outcomes confirm the methodology's efficacy under various test circumstances. As a result, the conversations validate the suggested work's strength and usefulness.

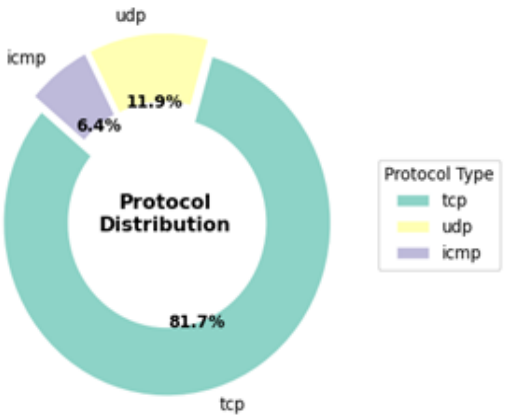

Figure 5 *Protocol Type Distribution*

The TCP protocol carries the majority of the traffic, making up 81.7% of the distribution, as seen in Figure 5. UDP has a moderate presence in the sample, accounting for 11.9% of the traffic. At 6.4%, ICMP has the lowest percentage, suggesting that it is not as widely used as other protocols. The dominance of TCP-based communication in the examined network traffic is demonstrated by this distribution.

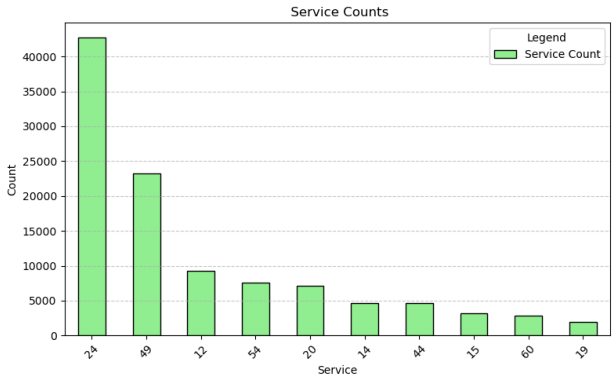

Figure 6 *Distribution of Service Counts in the Dataset*

Figure 6 represents, service 24 is the most frequent, with over 40,000 counts, followed by service 49, which has over 23,000 counts. Moderate frequencies, ranging from 6,000 to 9,000, are displayed by other services like 12, 54, and 20. Below 5,000 counts, the remaining services 14, 44, 15, 60, and 19 occur at significantly reduced levels. This distribution shows how some services dominate while the majority are comparatively less common.

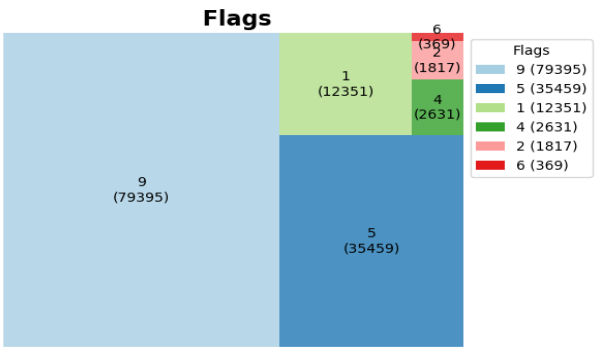

Figure 7 *Distribution of Flags in the Dataset*

With 79,395 occurrences, flag value 9 dominates the dataset, followed by flag 5 with 35,459 instances, as seen in Figure 7. Moderate frequencies of 12,351 and 2,631 are displayed by flag values 1 and 4, respectively. Flag values 2 and 6 on the other hand, with 1,817 and 369 counts, are extremely uncommon. This distribution shows that there is a significant disparity across the dataset's various flag categories.

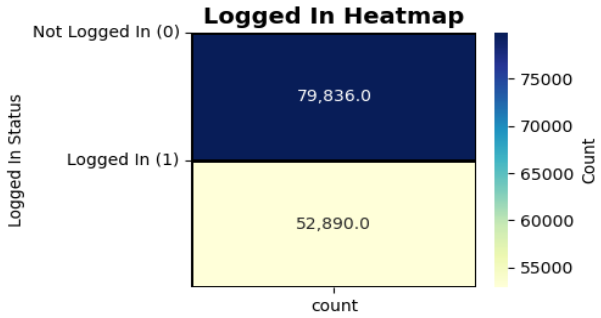

Figure 8 *Distribution of Logged-In Status*

With 79,836 entries, Figure 8 shows that there are more instances of people not being logged in in the dataset. By contrast, the logged-in status represents a smaller percentage, accounting for 52,890 records. This disparity draws attention to an imbalance between connections that are logged in and those that are not. Comprehension user activity patterns in the dataset requires a comprehension of this distribution.

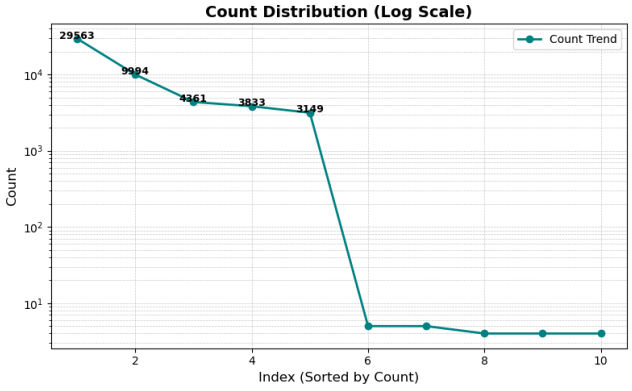

Figure 9 *Count Distribution of Classes on Log Scale*

Figure 9 demonstrates that the class distribution is highly imbalanced, with the first category having the highest count of 29,563. The next few classes gradually decrease in frequency, such as 9,994, 4,361, 3,833, and 3,149. Beyond this point, the remaining classes drop sharply to very low counts,

falling below 10. This skewed distribution highlights the dominance of a few classes while most categories are underrepresented.

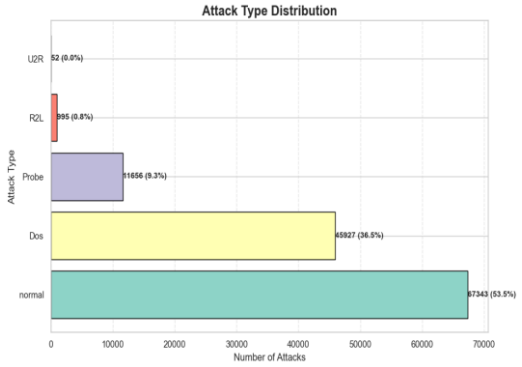

Figure 10 *Attack Type Distribution*

Figure 10 presented that the distribution of different attack types along with normal traffic in the dataset. It shows that normal records constitute the majority with 53.5%, followed by DoS attacks at 36.5%. Probe attacks account for 9.3%, while R2L and U2R represent only 0.8% and 0.0% respectively. This indicates a significant imbalance in attack categories, with DoS dominating among malicious activities.

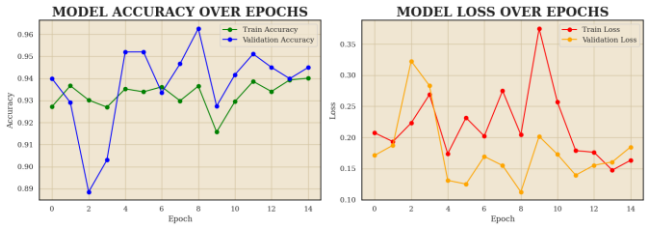

Figure 11 *Model Accuracy and Loss*

Figure 11 demonstrates that across all epochs, the model accuracy for both training and validation stays continuously over 90%. Although it varies more than training accuracy, validation accuracy often exhibits a similar pattern.

```
1181/1181 ━━━━━━━━━━━━━━━━━━━━ 1s 861us/step
Classification Report:
              precision    recall  f1-score   support

      normal     0.9874    0.9768    0.9821     13825
         Dos     0.8842    0.9794    0.9294      3540
       Probe     0.2235    0.9847    0.3643       327
         R2L     0.0464    0.4118    0.0833        17
         U2R     0.9844    0.9118    0.9467     20083

    accuracy                         0.9423     37792
   macro avg     0.6252    0.8529    0.6611     37792
weighted avg     0.9691    0.9423    0.9526     37792
```

Figure 12 *Model Classification report*

Figure 12 shows that the model accomplished an overall accuracy of 94.23%, indicating strong performance. The classes "normal," "DoS," and "U2R" achieved high precision, recall, and F1-scores. However, the model struggled with "Probe" and especially "R2L," showing very low precision and F1-scores. This highlights class imbalance issues, where minority classes are harder to predict accurately.

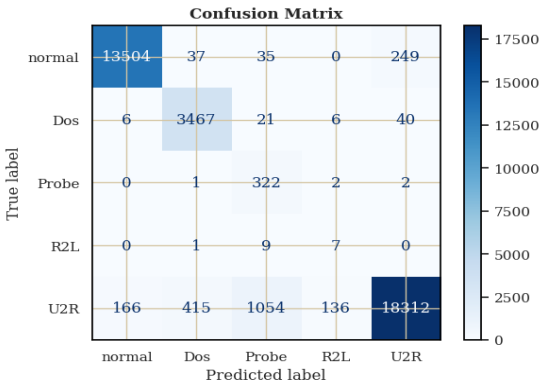

Figure 13 *Confusion Matric*

Figure 13 shows that the confusion matrix illustrates the classification performance across different attack types. The model correctly classified most "normal," "DoS," and "U2R" samples with minimal misclassifications. However, significant misclassification occurred for "Probe" and "R2L," where many instances were confused with other classes. This indicates strong performance on majority classes but weaker detection for minority attack categories.

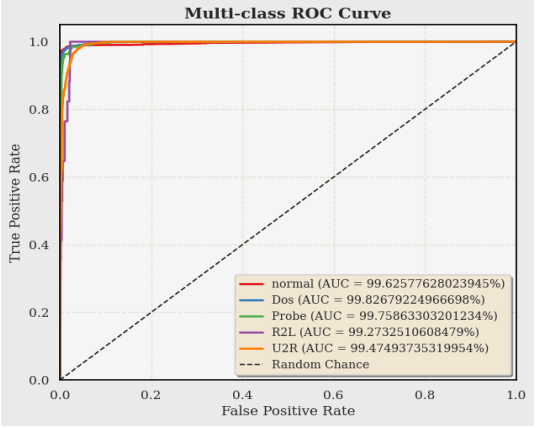

Figure 14 *Divergent ROC Curve*

Figure 14 illustrates that the Divergent ROC Curve demonstrates the system's ability to

distinguish between different attack types. All classes achieved very high AUC values, above 99%, indicating excellent classification performance. This confirms that the model performs robustly across all categories with strong discriminative power.

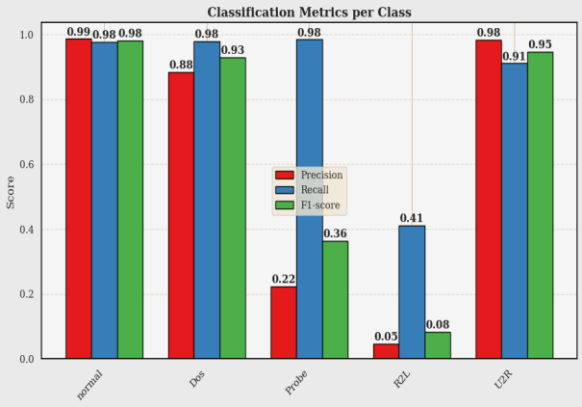

Figure 15 *Classification Metrics per Class*

Figure 15 shows that the classification metrics vary significantly across different classes. The "normal," "DoS," and "U2R" classes achieved high precision, recall, and F1-scores, indicating strong performance. In contrast, "Probe" and especially "R2L" had very low precision and F1-scores despite relatively better recall. This highlights the class imbalance problem, where minority attack types are harder to classify accurately.

Comparative Performance Analysis of the Proposed Method

Table 1: *Comparative Analysis*

Method	Accuracy
Decision Tree and Self-attentive Model [16]	80.25%
KNN [17]	76.5%
ResNet-BiLSTM	96%

The accuracy values of 3 different classification methods are compared in Table 1. The Decision Tree combined with the Self-attentive Model achieved an accuracy of 80.25% [18], which is higher than KNN, which obtained 76.5% [19]. However, the proposed ResNet-BiLSTM model significantly outperformed both with an accuracy of 96%. This clearly indicates that the suggested deep learning approach delivers more reliable and superior results compared to the other traditional methods.

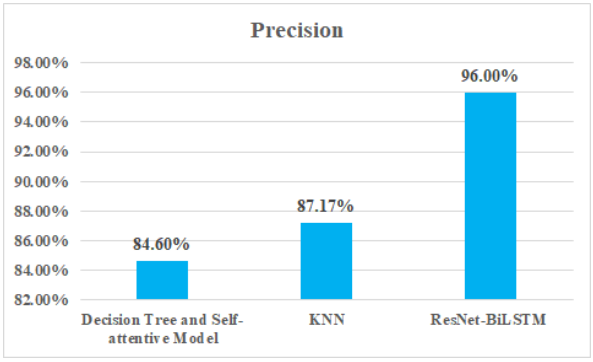

Figure 16 *Comparison of Precision*

Figure 16 shows that the precision comparison among different models demonstrates that the proposed ResNet-BiLSTM achieves the highest precision of 96.00%, clearly outperforming the other methods. The KNN model performs moderately with a precision of 87.17%, while the Decision Tree with Self-attentive Model records the lowest precision of 84.60%. This comparison highlights that ResNet-BiLSTM provides superior accuracy and reliability, making it more effective than traditional models for classification tasks.

5. Conclusion

This paper presents an improved ResNet-BiLSTM model for efficient intrusion detection. The system guarantees thorough assessment with strong pretreatment processes such as correcting values that are absent, Min-Max normalization by using the benchmark NSL-KDD dataset. In order to overcome class imbalance and improve the relevance of input characteristics, the Fennec Fox Optimization Algorithm (FFOA) is integrated for feature selection. Superior detection performance results from the enhanced ResNet-BiLSTM's ability to track temporal relationships and spatial correlations. The efficacy of the model is confirmed by the experimental findings, which show high precision (0.96), recall (0.94), F1-score (0.95), and accuracy (0.94). The Python-implemented system provides a dependable and scalable solution for contemporary intrusion detection situations while also lowering false

alarms. The framework is appropriate for real-time applications since it also exhibits resilience when managing massive amounts of data. Its adaptability ensures reliable performance even against evolving and sophisticated cyber threats. Overall, the proposed approach offers both practical applicability and significant improvements over traditional detection methods.

References

1. G. Logeswari; Swapneel Bose; T. J. I. A. Anitha, Year: 2023, “An intrusion detection system for sdn using machine learning”, Intelligent Automation & Soft Computing, Vol: 35, No: 1, pp. 867-880.

2. Murtaza Ahmed Siddiqi; Wooguil Pak, Year: 2022, “Tier-based optimization for synthesized network intrusion detection system”, IEEE Access, Vol: 10, pp. 108530-108544.

3. V. Raji; S. Maheswari; S. Sree Dharinya; P. Chinniah; Kavin KS; Kiruba Thangam Raja, Year: 2024, “An Effective Detection of Brain Tumor Detection Using ResNet-50 Model”, In 2024 International Conference on Advancement in Renewable Energy and Intelligent Systems (AREIS), pp. 1-6. IEEE.

4. V. Veerakumaran; S. Gokulprasanth; V Kesavan, Year: 2024, “Intrusion detection system (IDS) in networks”, Journal of Nonlinear Analysis and Optimization, Vol: 15, No. 4, ISSN: 1906-9685.

5. ATR Krishna Priya; A. Yazhini, “Integrating Whale Optimization Algorithm and Machine Learning for Efficient Traffic Management”.

6. Rachid Tahri; Youssef Balouki; Abdessamad Jarrar; Abdellatif Lasbahani, Year: 2022, “Intrusion detection system using machine learning algorithms”, In ITM Web of Conferences, Vol: 46, pp. 02003. EDP Sciences.

7. A. Bindhu; A. Ahilan; S. Vallisree; P. Maria Jesi; B. Muthu Kumar; Nikhil Kumar Marriwala; Aznul Qalid Md Sabr, Year: 2023, “Skin Cancer Diagnosis Using High-Performance Deep Learning Architectures”, In International Conference on Emergent Converging Technologies and Biomedical Systems, pp. 693-703. Singapore: Springer Nature Singapore.

8. K. Anusha; A. Ahilan; N. Muthukumaran; G. Muneeswari; A. Bhuvanesh; P. Maria Jesi, Year: 2025, “Trust and affinity based clustering for deterministic multicast routing using honey badger algorithm”, IETE Journal of Research, Vol: 71, No: 1, pp. 102-113.

9. Alaa Mohammed Banaamah; Iftikhar Ahmad, Year: 2022, “Intrusion detection in IoT using deep learning”, Sensors, Vol: 22, No: 21, pp. 8417.

10. Jia Huang; Zhen Chen; Sheng-Zheng Liu; Hao Zhang; Hai-Xia Long, Year: 2024, “Improved intrusion detection based on hybrid deep learning models and federated learning”, Sensors, Vol: 24, No: 12, pp. 4002.

11. Mohamed Abd Elaziz; Mohammed AA Al-qaness; Abdelghani Dahou; Rehab Ali Ibrahim; Ahmed A. Abd El-Latif, Year: 2023, “Intrusion Detection Approach for Cloud and IoT Environments Using Deep Learning and Capuchin Search Algorithm”, Advances in Engineering Software, pp. 176: 103402.

12. U. Yogish Pai; K. Krishna Prasad, Year: 2025, “AI-Driven Threat Intelligence for Predicting Advanced Persistent Attacks in Cloud-Based IT Services”, Systems Engineering and Management, Vol: 10, No: 51s.

13. Asmaa Halbouni; Teddy Surya Gunawan; Mohamed Hadi Habaebi; Murad Halbouni; Mira Kartiwi; Robiah Ahmad, Year: 2022, “CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System”, in IEEE Access, Vol: 10, pp. 99837-99849.

14. Vanlalruata Hnamte; Hong Nhung-Nguyen; Jamal Hussain; Yong Hwa-Kim, Year: 2023, “A Novel Two-

Stage Deep Learning Model for Network Intrusion Detection: LSTM-AE", in IEEE Access, Vol: 11, pp. 37131-37148.

15. Zhen Wang; Fuad A. Ghaleb, Year: 2023, "An Attention-Based Convolutional Neural Network for Intrusion Detection Model", in IEEE Access, Vol: 11, pp. 43116-43127.

16. Rachid Ben Said; Zakaria Sabir; Iman Askerzade, Year: 2023, "CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking with Hybrid Feature Selection", in IEEE Access, Vol: 11, pp. 138732-138747.

17. Shiming Li; Guangzhao Chai; Yuhe Wang; Guohui Zhou; Zhenxing Li; Dan Yu; Rencai Gao, Year: 2023, "CRSF: An Intrusion Detection Framework for Industrial Internet of Things Based on Pretrained CNN2D-RNN and SVM", in IEEE Access, Vol: 11, pp. 92041-92054.

18. Yuchen Lan; Tram Truong-Huu; Jiyan Wu; Sin G. Teo, Year: 2022, "Cascaded Multi-Class Network Intrusion Detection With Decision Tree and Self-attentive Model", 2022 IEEE International Conference on Data Mining Workshops (ICDMW), Orlando, FL, USA, pp. 1-7.

19. Pradeep Pandey, Year: 2024, "A KNN-Based Intrusion Detection System for Enhanced Anomaly Detection in Industrial IoT Networks", International Journal of Innovative Research in Technology and Science, Vol: 12, No: 6, pp.1-7.