WJCMS

Wasit Journal of Computer and Mathematics Science

Journal Homepage: https://wjcm.uowasit.edu.iq/index.php/WJCM

e-ISSN: 2788-5879 p-ISSN: 2788-5879

Enhancing Intrusion Detection with LSTM Recurrent Neural Network Optimized by Emperor Penguin Algorithm

Saif Wali Ali Alsudani[1,*] and Adel Ghazikhani[2]

[1]Iraqi Ministry of Justice, Baghdad, Iraq

[2]Imam Reza International University, Mashhad, Iran

*Corresponding Author: Saif Wali Ali Alsudani

DOI: https://doi.org/10.31185/wjcms.166https://doi.org/10.31185/wjcms.166

Received: June 2023; Accepted: September 2023; Available online: September 2023

ABSTRACT: Intrusion detection systems (IDS) have been developed to identify and classify these attacks in order to prevent them from occurring. However, the accuracy and efficiency of these systems are still not satisfactory. In previous research, most of the methods used were based on ordinary neural networks, which had low accuracy. Therefore, this thesis, with the aim of presenting a new approach to intrusion detection and improving its accuracy and efficiency, uses long-term memory (LSTM) optimized with the Penguin optimization algorithm (EPO). In the proposed approach, first, the features were pre-processed by normalization, cleaning, and formatting in number format. In the next step, the linear discriminant analysis (LDA) method was used to reduce the dimensions of the processed features, and after that, the EPO algorithm was used to optimize the size of the hidden unit of the LSTM network. Finally, the optimized network was evaluated using the NSL-KDD dataset, which is a widely used benchmark dataset in the field of intrusion detection. The results obtained for the training and test datasets were 99.4 and 98.8%, respectively. These results show that the proposed approach can accurately identify and classify network intrusions and outperform many existing approaches.

Keywords: Intrusion Detection Systems, Penguin Meta-Heuristic Algorithm, Long-Term Memory Neural Network, Linear Detection Analysis.

1. INTRODUCTION

In our digital landscape, escalating cyber-attacks emphasize robust intrusion detection systems (IDS) criticality. The surge in threats targeting data, continuity, and reputation intensifies the need for effective IDS. These systems vigilantly monitor activities, and swiftly identify and counteract malicious actions, upholding digital integrity [1]. Deep learning, especially Long Short-Term Memory (LSTM) networks, proves transformative in domains like cybersecurity. LSTM's capacity to process time-series data aids in pattern extraction from network traffic [2]. LSTM networks, a kind of recurrent neural network (RNN), capture and retain data over sequences, excelling at sequence prediction, natural language processing, and intrusion detection [3]. The Emperor Penguin Algorithm (EPA) draws inspiration from penguin colonies in Antarctica, blending exploration and cohesion. This underpins the EPA's concept [4]. EPA aids optimization, mirroring penguin adaptability via dynamic agent-environment interactions. EPA navigates exploration-exploitation trade-offs, akin to penguins' adaptive strategies [5]. The research aims to optimize LSTM with EPA for intrusion detection. It explores EPA's dynamic balance to enhance LSTM training and performance in network traffic analysis [6]. This study aims to enhance IDS accuracy, efficiency, and adaptability, integrating EPA's collective intelligence into LSTM optimization. Advancing cybersecurity against evolving threats is the goal [7].

2. LITERATURE REVIEW

Conventional machine learning methods, like SVM [8], [9], KNN [10], ANN [11], RF [12], [13], and others [14], [15], have found success in earlier intrusion detection research. However, studies highlight deep learning's superiority, propelling its adoption for intrusion detection. Experiment results [16] demonstrate deep learning's effectiveness in flow-based anomaly detection in software-defined networks. The authors employ a deep neural network-based technique. In another study [17], deep learning with self-taught learning (STL) enhances network intrusion detection using the NSL-KDD dataset. Deep learning's capacity to reduce feature dimensions is a key focus. Pre-training mostly uses deep learning, conducting classification with conventional supervision. Direct classification is rare, and multiclass effectiveness is unexplored. Reduced-size neural networks (RNNs), as in [18], lack deep learning's high-dimensional feature representation capacity. Authors propose a 3-layer RNN for misuse-based IDS with limited nodes and no binary classification analysis.

The surge in cyber threats emphasizes the need for artificial intelligence and machine learning-based breach detection. The LSTM network excels in sequential data modeling, yet optimizing it for intrusion detection, particularly hidden layer count, poses challenges. Addressing this, the Penguin Optimization Algorithm (POA) fine-tunes LSTM's hidden layer count. POA draws from penguin social behavior, efficiently navigating the solution space. Experimental results show heightened accuracy with POA-enhanced LSTM, reducing false positives and negatives. Leveraging POA to optimize LSTM's hidden layer count in intrusion detection showcases a pioneering approach for enhanced accuracy and performance.

3. PROPOSED FRAMEWORK

Network intrusion detection is vital for cybersecurity, aiding real-time threat recognition. Systems fall into two types: signature-based (using known attack patterns) and anomaly-based (ML detecting abnormal behavior). A potent ML algorithm is a short-long-term memory network (LSTM) – that excels at learning sequential data dependencies. The Penguin algorithm, inspired by penguins' search behavior, optimizes LSTM via hive-simulated penguin behavior. Our innovation: Penguin optimizes LSTM hyperparameters (e.g., hidden layers) for intrusion detection. This tuned LSTM via Penguin improves accuracy and efficiency, enhancing cybersecurity against attacks. This tech promises breakthroughs in intrusion detection, reinforcing cloud security.

3.1 STEPS OF THE PROPOSED METHOD

As we mentioned in the introduction, the innovative aspect of this thesis is to detect intrusion or normal traffic in the network, using the LSTM network and the Penguin algorithm to optimize the number of hidden layers of this network. The steps of the proposed model are detailed in the following flowchart:

3.2 DATA SET PREPROCESSING IN MACHINE LEARNING (ML)

Data preprocessing is crucial in ML, converting raw data into a usable format for algorithms. It involves cleaning, transforming, and structuring data for easy analysis. Preprocessing significantly impacts model performance; data quality and structure affect accuracy and reliability. Inadequate preprocessing hampers pattern recognition and prediction, yielding unreliable models. It also curbs overfitting, where models overly adapt to training data, failing on new data. Overall, data preprocessing ensures accurate, reliable models. It includes cleaning, normalizing, and integrating data for the model's suitability. Common techniques:

3.2.1. DATA CLEANING IN ML

Data cleaning iteratively corrects errors in a dataset. The aim is accuracy, consistency, and error-free data for machine learning. Clean data yields precise, reliable predictions. Techniques include mean/mode for missing values and stats for noise. Here, missing values were replaced with the average of adjacent samples. Nearby missing data neighbors used the average for replacement.

3.2.2. Data Normalization in ML

Normalization scales numerical data to a standard range. Its goal is equitable feature influence during training, crucial for varied-scale features. Common scale, e.g., 0 to 1, improves ML performance. Standardization and min-max scaling are techniques used; here, min-max scaling is employed, calculated as:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

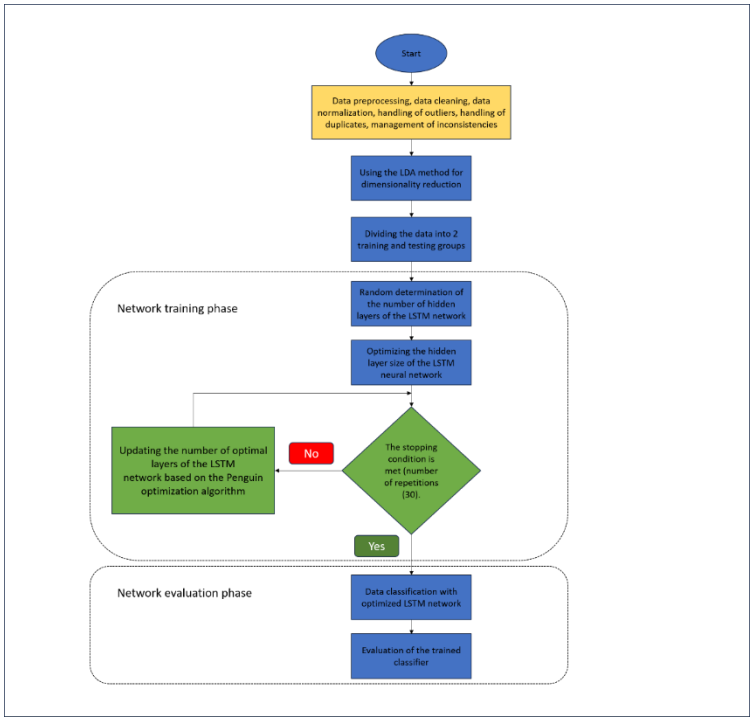

FIGURE 1. Process flowchart of the proposed model

3.2.3. OUTLIER MANAGEMENT IN ML

Outliers are extreme data points deviating from the dataset. They impact ML results, causing skewed distribution and biased predictions. Variance is affected, hampering model generalization. If caused by errors, outlier removal is considered.

3.2.4. HANDLE DUPLICATES

Duplicates stem from collection or entry errors. They impair the model performance, necessitating pre-training removal.

3.2.5. INCONSISTENCY MANAGEMENT IN ML

Inconsistencies are addressed in data processing for ML. It detects and corrects data errors, inconsistencies, and anomalies. Incorrect data may arise from entry, processing, or convergence errors. Such inconsistencies diminish model accuracy. Collecting data from diverse sources/methods can introduce conflicts. Managing inconsistencies is vital for model accuracy. Techniques like data normalization handle these issues.

3.2.5.1. REDUCTION VIA LINEAR DISCRIMINANT ANALYSIS (LDA. Clean data is available, but high-dim. datasets are resource-intensive and prone to overfitting. Linear Discriminant Analysis (LDA) reduces dimensions and separates classes.

LDA is a stats technique for dim. reduction and classification in ML. It's supervised, aiming to find linear combos of features for class separation. Data is projected into lower dimensions, maximizing the between-class vs. within-class variance ratio. Useful for feature extraction when features > observations. It boosts ML model accuracy and efficiency by reducing data dimensions. LDA not only reduces dim. but also maximizes class separation. It's particularly effective for correlated/noisy data. LDA is powerful for feature extraction, dim. reduction, and classification in ML. It's valuable for enhancing model accuracy/efficiency by reducing input dim. and optimizing class separation.

3.2.6. DATA SEGMENTATION: TRAINING & TESTING IN ML

Data is often split into training and test sets in ML. Goal: Assess the model on new data. Training set fits model; test set evaluates its performance on unseen data. The test set mirrors real-world data kept separate from training. Common: random split, e.g., 80% train, 20% test. Methods like stratified sampling maintain class proportions. Random split impacts model performance, addressed by Hold_out cross-validation. Segmenting data into training/testing is vital, gauging model

performance on new data and avoiding overfitting. Split percentage and technique influence model performance.

3.3 DATA CLASSIFICATION WITH LSTM NETWORK OPTIMIZED BY PENGUIN ALGORITHM

In the subsequent section, we classify network data to detect intrusion or normal traffic. This involves using a Long Short-Term Memory (LSTM) network, a recurrent neural network variant. LSTM excels in handling sequential data like time series, speech, and text, addressing the vanishing gradient issue of conventional RNNs. It employs a memory cell to retain information over time while updating selectively.

An LSTM network includes crucial components: input gate (storage control with sigmoid activation), forget gate (removes data with sigmoid activation), memory cell (stores modifiable information), and output gate (determines LSTM output using sigmoid and hyperbolic tangent). The LSTM refines its weights during training via a loss function like mean squared error or cross-entropy, enabling accurate predictions on sequential data. Interconnected gates create a short-term memory mechanism, including:

1. Input Gate: Manages memory cell storage, evaluating current input to decide retention (0 to 1).
2. Forget Gate: Controls memory cell data removal, considering prior output (sigmoid).
3. Memory Cell: Stores information, subject to selective updates or forgetfulness.
4. Output Gate: Determines LSTM output (sigmoid and hyperbolic tangent.

Throughout the training, LSTM optimizes weights for accurate predictions.

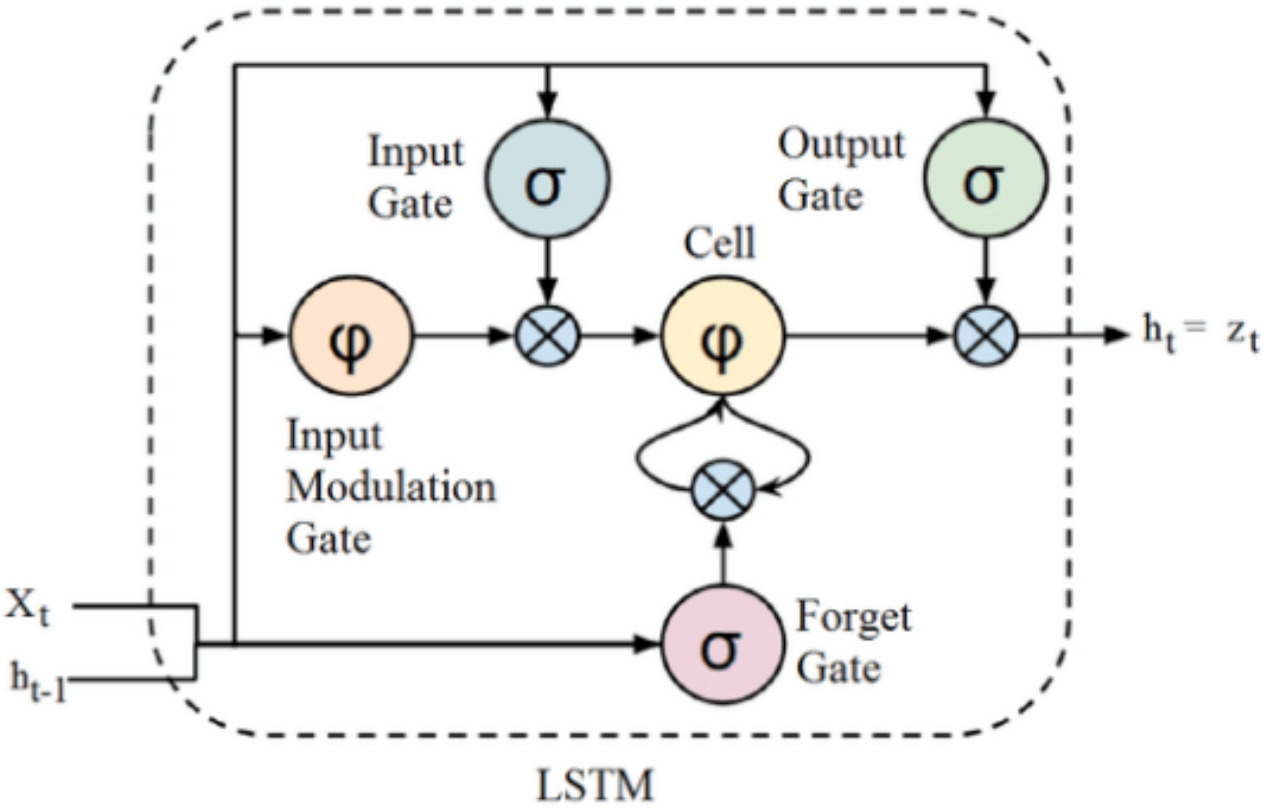

FIGURE 2. LSTM network structure and its gates

The gate-regulated LSTM network utilizes adjustable parameters updated through backpropagation during training. Sigmoid activations control input, forget, and output gates, impacting data retention. A hyperbolic tangent function manages memory cell values. These mechanisms enable long-term data dependency capture. LSTM, a key recurrent neural network, excels in sequential data analysis. Its memory cell addresses gradient vanishing. Hidden layers within

LSTM record input sequence dependencies; layer size affects this. Balancing capacity and overfitting, optimal size is crucial. Penguin optimization, inspired by penguin behavior, optimizes hidden layer size. It mimics population-based search with social and individual learning, preventing local optima.

Parameter adaptation: Adjust parameters for the problem - population, repetitions, search space, objective function.

Key factors: Initial penguin population impacts solution candidates; Objective function minimizes error for optimal parameter values.

$$Errors = 1 - Accuracy$$

- Exploration Speed: Balances exploration and exploitation; a higher rate increases exploration but may slow convergence.
- Termination Condition: Sets max iterations for termination; more aids exploration but raises computational costs.
- Optimal Solution Selection: In POA, minimizes objective function; the lowest value is the best solution.
- Next Step: Combining LSTM NN with Penguin optimization for improved network intrusion detection classification.

4. RESULTS OF THE EXPERIMENT

In this research, simulation results and the implementation of the proposed method in MATLAB software are detailed, and aligned with the specifications in Table 1. The method is fully modeled, including efforts to optimize LSTM neural network hyperparameters via the Emperor Penguin Optimization Algorithm for network intrusion detection. Pre-processing techniques like normalization and text analysis, as well as LDA, are applied to prepare data and reduce input dimensions. We structured it into four main sections. The second part introduces the input dataset. The third part outlines evaluation criteria for the proposed model, encompassing precision, accuracy, recall, and F1 score – all assessing model performance. The fourth section presents simulation results using the proposed model. The method is executed comprehensively, and individual section results are scrutinized. Graphical and statistical displays of outcomes, such as scatter matrices, ROC diagrams, and performance-related graphs, are included. In the concluding section, a comparison is drawn between the proposed method and other existing intrusion detection methods.

Here, merits and drawbacks are examined, and the comparative outcomes are reported in detail.

Table 1. Specifications of the system used

Item	Specifications
System model	HP Pavilion dv6 Notebook PC
Processor	Intel Core i7 (3610QM) @ 2.3 GHz
Random memory	(8 GB)
Hard disk	(500 GB SSD)
Operating system	Windows 10 Pro 64-bit

4.1 DATA COLLECTION

In this study, the NSL KDD dataset [19] is employed for training and evaluating the LSTM model. The NSL KDD dataset is an updated version of the 1999 KDD Cup dataset, aiming to rectify the previous dataset's unrealistic portrayal of network traffic. This dataset is categorized into two classes: attack and normal traffic, representing network traffic data within a simulated environment. It encompasses 41 features derived from network packets, encompassing attributes like connection duration, protocol type, source and destination addresses, and more. Within these 41 features, three are symbolic, while the rest are continuous. Symbolic features lack numerical values and need to be transformed into a numerical and interpretable format for utilization in machine learning algorithms. For a comprehensive view of the features within the NSL KDD dataset, refer to Table 2.

4.2 EVALUATION CRITERIA

Evaluation criteria are pivotal in decision-making processes, offering a structured framework to assess differing choices. They empower decision-makers to opt for the most fitting path using accessible parameters and data. This study examines fundamental evaluation metrics, including precision, accuracy, recall, and the F1 score. Herein, each metric is elucidated:

Table 2. Attributes of the NSL-KDD Dataset

Number	Data Features	Number	Data Features
1	Duration	22	Is_guest_login
2	Protocol_type	23	Count
3	Service	24	Srv_count
4	Flag	25	Serror_rate
5	Src_bytes	26	Srv_serror_rate
6	Dst_bytes	27	Rerror_rate
7	Land	28	Srv_rerror_rate
8	Wrong_fragment	29	Same_srv_rate
9	Urgent	30	Diff_srv_rate
10	Hot	31	Srv_diff_host_rate
11	Num_failed_logins	32	Dst_host_count
12	Logged_in	33	Dst_host_srv_count
13	Num_compromised	34	Dst_host_same_srv_rate
14	Root_shell	35	Dst_host_diff_srv_rate
15	Su_attempted	36	Dst_host_same_src_port_rate
16	Num_root	37	Dst_host_srv_diff_host_rate
17	Num_file_creations	38	Dst_host_serror_rate
18	Num_shells	39	Dst_host_srv_serror_rate
19	Num_access_files	40	Dst_host_rerror_rate
20	Num_outbound_cmds	41	Dst_host_srv_rerror_rate
21	Is_host_login		

Accuracy: This metric gauges overall model performance by calculating the ratio of correctly classified instances to the total instances. In intrusion detection, accuracy evaluates the model's capability to accurately discern normal network traffic from intrusion. However, accuracy is cautioned against in unbalanced class scenarios.

Precision: Precision gauges the ratio of true positive predictions to the total positive predictions made by the model. Within intrusion detection, it measures the proportion of correctly identified intrusion instances among all predicted intrusions.

Recall: Recall quantifies the ratio of true positive predictions to the total true positive instances. In intrusion detection, recall signifies the model's capacity to correctly identify all intrusion instances.

F1 Score: This metric harmonizes precision and recall, presenting an overarching measure of model performance. It calculates the harmonic mean of precision and recall, proving advantageous in situations with imbalanced classes.

In essence, these metrics provide a robust basis for evaluating and enhancing model performance, aiding decision-making processes in various applications, including intrusion detection.

4.3 RESULTS ANALYSIS

This section thoroughly evaluates the proposed method, focusing on step-by-step analysis. Initial attention centers on dataset pre-processing. Due to the NSL-KDD dataset's missing samples and character-oriented nature, multiple transformations occur. Character data is tokenized and converted into numbers while missing data is estimated using neighboring averages. Labels are separated, and input data is normalized.

Note that some features have uniform values across samples. During normalization, values can become NaN due to fixed features. Removing such features is justified as they lack classification insights without complications. Thus, 38 features remain.

Yet, this input dimension is still substantial, raising redundancy and overfitting concerns. Linear Discriminant Analysis (LDA) is used. LDA, a supervised technique, reduces dimensionality while highlighting class separations. By computing dispersion matrices, eigenvalues and eigenvectors are derived. Utilizing eigenvectors, 20% of dimensions are separated, resulting in 8 features for final processing. This strategic reduction improves performance and reduces overfitting. Figure 3 shows enhanced data separability post-LDA.

This analysis demonstrates the proposed approach's effectiveness and LDA's role in feature enhancement.

4.4 COST FUNCTION EVALUATION & OPTIMIZATION STRATEGY

The assessment of this cost function entails time-intensive LSTM network training. However, the search space is one-dimensional, confined in dimensionality, leading to swift convergence. To expedite optimization, parameters are configured thoughtfully. Algorithm iterations are capped at 12, while population members number 8. During optimization, network training prioritizes speed over accuracy. Notably, optimization seeks comparison solely, enabling swift calculations without sacrificing final accuracy. Rigorous training is reserved for the end phase.

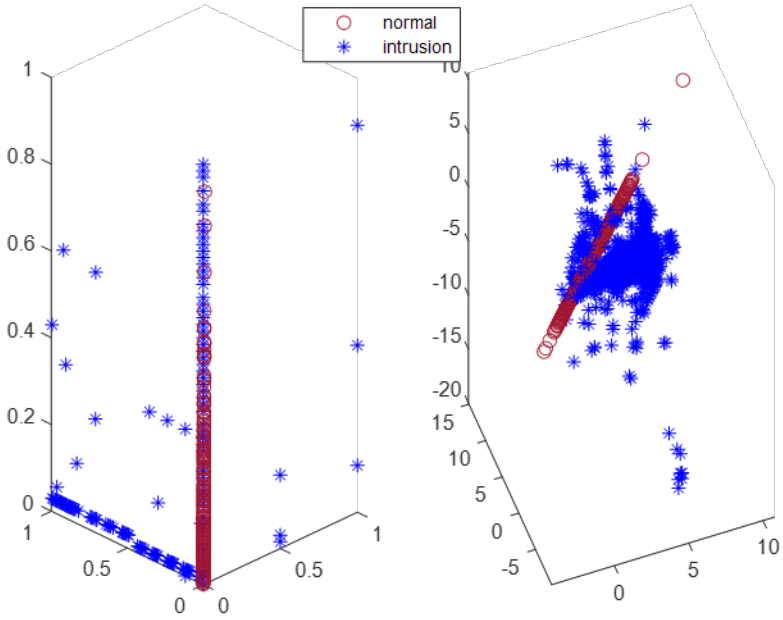

FIGURE 3. Separation of data before and after linear discriminant analysis

Figure 4 visually depicts the Emperor Penguin optimization algorithm's convergence journey. Post determining hidden units (Table 3), network training initiates, executed 100 times with a batch size of 20. Figure 5 illustrates the LSTM network training progression. Subsequent phases involve a comprehensive evaluation of the trained network using diverse criteria and graphical representations.

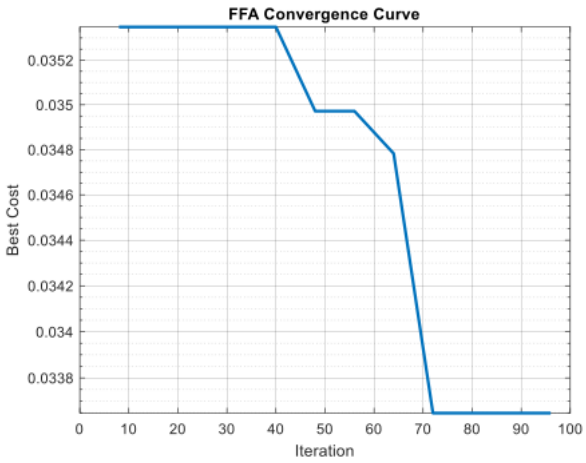

FIGURE 4. Convergence Diagram of POA in the Process of Determining Hyperparameters of CNN

Table 3. Optimal CNN Hyperparameter Values in the Problem

	Number of hidden layers
Value	83

Finally, after calculating the optimal parameters, we train the convolutional neural network with much stricter settings using the ADAM training algorithm, The number of repetitions is 100, and the batch size is equal to 10. Figure 5 shows

Table 4. The general structure of the confusion matrix in the intrusion detection problem

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

the process of training the mentioned network. In the following, we will examine the performance of this trained network more closely in order to comprehensively examine the proposed approach.

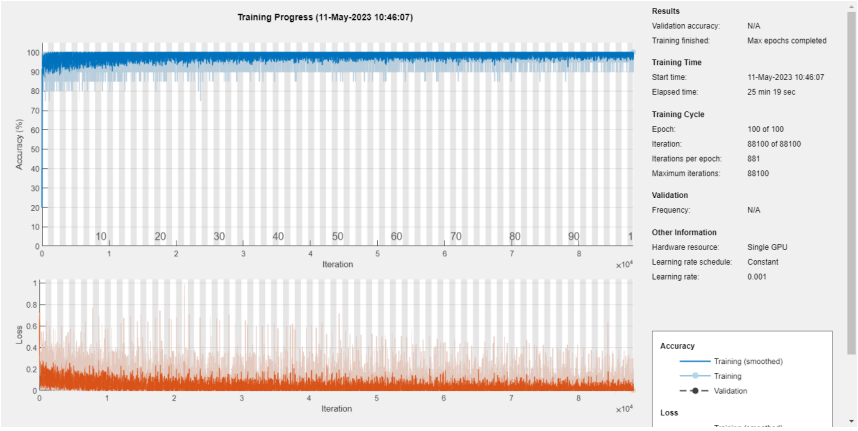

FIGURE 5. Convolutional neural network training process

4.5 EXAMINING RESULTS OF TEST AND TRAINING SETS BASED ON THE CONFUSION MATRIX

Scrutinizing model performance is pivotal in results analysis and evaluation. The confusion matrix emerges as a potent tool, furnishing a holistic understanding of network performance. This matrix presents a tabular representation, showcasing correctly and incorrectly classified instances through parameters including True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The matrix proves highly beneficial in binary and two-class scenarios. Thus, in this study, we deploy the confusion matrix to assess the LSTM classifier model, distinguishing between "infiltration" and "normal traffic".

The four parameters within the matrix are delineated as follows:

- True Positive (TP): Accurate identification of "infiltrators".
- False Positives (FP): Incorrect classification of "normal traffic" as "infiltration".
- True Negative (TN): Precise identification of "normal traffic".
- False Negatives (FN): Incorrect classification of "intrusion" as "normal traffic".

The overall configuration of the confusion matrix is exemplified in Table 4, visually encapsulating the model's classification performance across the two defined classes.

In the following, with the help of the parameters defined in the confusion matrix, we will calculate the evaluation criteria mentioned in section 4.2:

1) $Accuracy = \frac{TP_y+TN_y}{TP_y+TN_y+FP_y+FN_y}$

2) $Precision = \frac{1}{n_c}\sum_y (\frac{TP_y}{TP_y+FP_y})$

3) $Recall = \frac{1}{n_c}\sum_y (\frac{TP_y}{TP_y+FN_y})$

4) $F_1 Score = \frac{2\times(Precision\times Recall)}{Precision+Recall}$

The confusion matrix of the mentioned method for the training and test datasets is shown in Figures 6 and 7, respectively:

FIGURE 6. Training clutter matrix

FIGURE 7. Test clutter matrix

The results show an error detection rate of 0.56% in training and 1.21% in testing for the new method. This suggests successful hyperparameter optimization and generalization of the LSTM network to new data.

The study evaluates model performance using ROC curves, assessing intrusion detection (IDS) on two classes: penetration and normal traffic. ROC plots TPR against FPR at various thresholds. Higher curve placement indicates accurate classification, with high TPR and low FPR. AUC-ROC, ranging 0-1, gauges classifier performance. Figures 8 and 9 display training and test ROC curves.

Using criteria from section 4.2, the model's performance on test and training sets is evaluated. Figures 10 and 11 depict these results. Accuracy is 99.44% in the test and 98.78% in training, with balanced values across criteria, indicating unbiased modeling. These outcomes suggest a promising future for intrusion detection with this method.

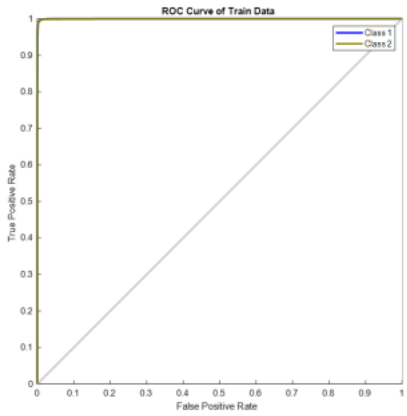

FIGURE 8. Performance characteristic curve of training data

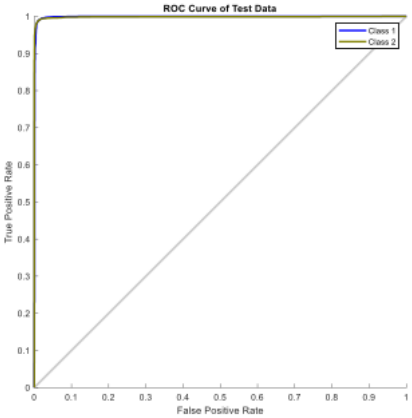

FIGURE 9. Test data characteristic curve

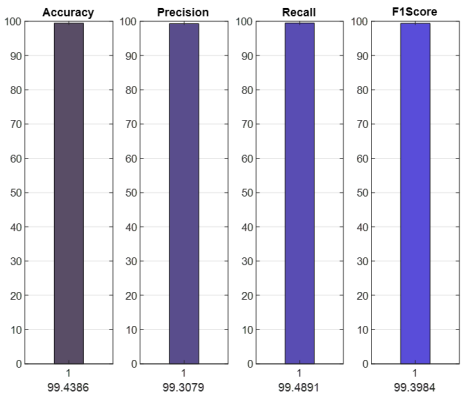

FIGURE 10. Final review of training data

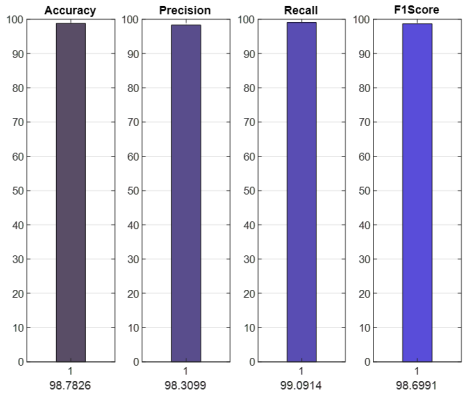

FIGURE 11. Final review of Test data

4.6 COMPARING RESULTS

We contrast our proposed technique in this work with prior research. Three references are analyzed:

In Ref. [20], a Recurrent Neural Network (RNN) is used to detect network infiltrations. This reference compares its approach to methods like Simple Bayes and Random Forest on the NSL-KDD dataset. Results demonstrate 98.5% accuracy in classifying penetration.

In Article [21], an algorithm enhances abnormal intrusion detection using the KDD'99 dataset. The proposed algorithm achieves 98.42% accuracy by fine-tuning support vector machine parameters through SA. Article [22] elevates attack classification accuracy by combining support vector machine, particle swarm, and WMV. Testing on the KDD99 dataset shows the proposed method reaches an 82.897% accuracy.

Our method employs a Penguin optimization-tuned short-long-term memory algorithm to classify intrusions in the NSLKDD dataset. Current research attains an accuracy of 98.78%, highlighting its superiority over existing studies.

4.7 DISCUSSION

The novel intrusion detection method employs Penguin-optimized LSTM networks to tackle cyber-attacks in network-reliant settings. Current IDS suffer accuracy and efficiency issues, necessitating innovative solutions. EPO algorithm enhances LSTM networks for better results. The method encompasses sequential steps, from preprocessing to EPO-optimized LSTM, emphasizing data quality for robust intrusion detection. Linear discriminant analysis optimizes resource use. Evaluated on NSL-KDD, it achieves 99.4% training and 98.8% test accuracy, surpassing rivals. This boosts reliable intrusion detection. The approach not only enhances accuracy but also reduces computation time, making it practical and efficient. Real-time use is feasible, thanks to LSTM networks capturing temporal dependencies crucial for complex attack detection.

5. CONCLUSION

The introduction of intrusion detection using LSTM networks optimized through the Penguin Optimization algorithm is a significant advancement in cybersecurity. This method effectively addresses accuracy and efficiency challenges in Intrusion Detection Systems, performing well on the NSL-KDD dataset. By combining preprocessing, LDA-driven dimension reduction, and EPO-optimized LSTM, it enhances accuracy and optimizes resources. Achievements on NSL-KDD showcase its potential to surpass existing techniques for real-time intrusion detection. The method's broad applicability, shown through a widely used dataset, encourages further exploration and comparison. As cyber threats evolve, this approach strengthens accuracy, efficiency, and overall effectiveness in securing network environments.

FUNDING

None

ACKNOWLEDGEMENT

None

CONFLICTS OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] J. Anderson *Cybersecurity Threats, Vulnerabilities, and Trends: An Overview*, 2022.
[2] H. Zhang, J. Liu, C. Wu, and J. Wang, "A Comprehensive Survey of Deep Learning in Network Anomaly Detection," *IEEE Access*, vol. 10, pp. 15359–15377, 2022.
[3] Y. Li, Y. Jin, and D. Gong, "Anomaly Detection for Industrial Control Systems Using Bidirectional LSTM with Attention Mechanism," *IEEE Transactions on Industrial Informatics*, 2023.
[4] X. Lin, X. Zheng, J. He, and D. Huang, "An Improved Emperor Penguin Optimization Algorithm for Solving Numerical Optimization Problems," *IEEE Access*, vol. 9, pp. 36753–36764, 2021.
[5] P. K. Sahu and S. Panda, "Hybridization of Emperor Penguin Algorithm with Differential Evolution for Solving Economic Load Dispatch Problem," *Swarm and Evolutionary Computation*, vol. 67, pp. 100997–100997, 2022.
[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 849–864, 2018.
[7] Y. Wang, C. Jia, K. Ren, and W. Lou, "Neural Network Ensembles for Intrusion Detection: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 2244–2272, 2021.
[8] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Appl. Soft Comput*, vol. 18, pp. 178–184, 2014.
[9] R. R. Reddy, Y. Ramadevi, and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1148–1153, 2016.
[10] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A new intrusion detection system based on KNN classification algorithm in wireless sensor network," *IEEE J. Elect. Comput. Eng*, vol. 2014, no. 240217, 2014.
[11] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," *Proc. IEEE Int. Conf. Signal Process*, pp. 92–96, 2015.
[12] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci*, vol. 89, pp. 213–217, 2016.
[13] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *Man, Cybern. C, Appl. Rev*, vol. 38, no. 5, pp. 649–659, 2008.
[14] J. 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.
[15] 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.
[16] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *presented at the 9th EAI Int. Conf. Bio-inspired Inf*, pp. 21–26, 2016.
[17] T. A. Tang, L. Mhamdi, D. Mclernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," *Proc. IEEE Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, pp. 258–263, 2016.
[18] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," *Neural Comput. Appl*, vol. 21, no. 6, pp. 1185–1190, 2012.
[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–6, 2009.
[20] C. Yin, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[21] S. W. Lin, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Appl. Soft Comput*, vol. 12, no. 10, pp. 3285–3290, 2012.
[22] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput*, vol. 38, pp. 360–372, 2016.