International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

An Efficient Feature Selection-based DDoS Attack Detection in Cloud Computing using Optimized LSTM

M.C Malini[1], N. Chandrakala[2]

[1]Department of Computer Science, SSM College of Arts & Science, Komarapalayam, Namakkal, India.
Email: *malinimarianesan[at]gmail.com*

[2]Department of Computer Science, SSM College of Arts & Science, Komarapalayam, Namakkal, India.
Email: *nchandrakala15[at]gmail.com*

Abstract: ***DDoS attacks are considered one of the most severe security risks to the cloud computing environment by the fact that it is capable of overloading resources, affecting the service availability. The dimensionality, redundancy, and time constraints of cloud network traffic are quite high, which complicates the use of traditional intrusion detection systems. To resolve these problems, the proposed paper will propose an effective DDoS attack detection model, which combines autoencoder-based feature selection and an optimized Long Short-Term Memory (LSTM) model. The autoencoder is used to learn in an automatic manner compact and discriminative feature representations of high-dimensional traffic data, and thus eliminate redundancy and enhance learning efficiency. In order to improve the performance of detection, the LSTM network is optimized by an Improved Firefly Algorithm (IFA), which is augmented by Partial Opposition-Based Learning (POBL). Diversification of the population is enhanced by the use of POBL, which also speeds up convergence, allowing a good tuning of hyperparameters without premature convergence. The optimized LSTM is very effective in capturing long-term temporal dependencies in the network traffic, which are necessary in the correct differentiation of DDoS attacks and normal cloud traffic. The proposed framework is tested on benchmark DDoS datasets frequently utilised in cloud security studies, and the performance of the framework is compared with traditional LSTM and alternative metaheuristic-optimised LSTM frameworks. The experimental findings indicate that the suggested method has high accuracy, precision, recall, and F-score, as well as a faster and more stable convergence rate. The results substantiate that the autoencoder-selected features, combined with IFA, POBL-optimized LSTM, give a solid, efficient, and scalable algorithm to detect DDoS attacks in real time in the context of cloud computing.***

Keywords: DDoS attack detection; hyperparameters optimization; firefly algorithm; long short-term memory; partial opposition-based learning; autoencoder

1. Introduction

Cloud computing has been an element of modern information technology by allowing computing resources and services to be accessed via the Internet and on a scale and on-demand. Its extensive implementation in fields like finance, healthcare, education, and e-commerce has greatly enhanced the efficiency and flexibility of operation. Nevertheless, the openness and shared environment of cloud infrastructures also predetermine their high vulnerability to cyber threats, including the ones that can be destructive, such as DDoS attacks. DDoS attacks aim at overwhelming cloud resources by causing massive amounts of malicious traffic, which causes degradation of services, denying the legit user access, and causing enormous financial losses. The growing complexity, diversification, and time sensitivity of DDoS attack patterns are very challenging to conventional detection controls.

The traditional DDoS detection methods, such as signature detection and classical machine learning methods, are not usually compatible with the dynamic and high-dimensional nature of cloud network traffic. The recent developments in deep learning have demonstrated encouraging outcomes in intrusion detection because they can learn complex data representations automatically. Nevertheless, deep learning models are extremely sensitive to superfluous and inappropriate features of raw traffic data that raise the computational load and adversely impact the accuracy of the detection. The use of autoencoders to select features has thus become a beneficial remedy to obtain compact and discriminative feature representations that are easy to reduce dimensionality, whilst maintaining important traffic properties.

The DDoS detection in the cloud environment is best achieved by LSTM networks as they are capable of obtaining long-term temporal dependencies and sequential patterns in network traffic flows. However, the effectiveness of LSTM models is determined by the choices of hyperparameters. Poor tuning may also result in a slow convergence, early stagnation in local optima, and low generalization ability. To solve this problem, optimization algorithms based on metaheuristics have been extensively used, and it is possible to mention the Firefly Algorithm due to its simple nature and global search capability. Nevertheless, traditional Firefly-based optimization algorithms might continue to exhibit low exploration and early convergence in high-dimensional, complex search spaces.

In order to address these shortcomings, the present study suggests an effective DDoS attack detecting framework that combines autoencoder-driven feature selection and an optimized LSTM model by use of an IFA with POBL. The partial opposition-based learning process enhances the diversity of the population and speeds up the convergence process as it takes into account partially opposite solutions in the process of optimization, resulting in more robust and

stable hyperparameter tuning. The main objective of this study is to create an effective and scalable cloud computing-based DDoS attack detection framework through an integration of autoencoder-based feature selection and an Improved Firefly Algorithm-based LSTM framework with Partial Opposition-Based Learning to improve the detection accuracy, convergence speed, and computational efficiency. This study will be important in the context of offering a powerful, precise, and computationally optimal DDoS identification solution specific to cloud computing appliances. Through the incorporation of an autoencoder-based selection of features with an IFA-LSTM model, the proposed framework will help solve problems associated with high-dimensional data, premature convergence, and temporal traffic analysis. The convergence behavior and the performance of the proposed approach are better, which makes it appropriate to real-time and large-scale deployments in clouds. Moreover, the suggested framework can be applied to additional intrusion detection applications and cybersecurity tools, which will result in the emergence of smart and secure cloud security systems. The main contributions of this work can be outlined in the following way:

- A viable autoencoder-based feature selection methodology is presented to remove redundant and irrelevant network traffic characteristics to achieve dimensional reduction and low complexity.
- The combination of Partial Opposition-Based Learning and the Improved Firefly Algorithm is used to come up with a novel optimization strategy that optimizes the hyperparameters of LSTM.
- A more streamlined LSTM-based DDoS detection model is suggested to capture the temporal relationships in the cloud network traffic.
- Intensive experimental tests are applied to benchmark datasets, and the suggested model is contrasted with classical and optimization-based LSTM methodologies in terms of standard performance measures.
- Convergence analysis is made to show that the proposed optimization framework is stable and efficient.

The other sections of the paper are organized in the following way: Section 2 presents and discusses the recent associated works related to detecting DDoS attacks in cloud computing with references to deep learning and optimization-based methods. Section 3 contains the proposed research methodology, optimized LSTM model and Improved Firefly Algorithm with Partial Opposition-Based Learning. Section 4 explains the experimental design and the performance analysis and the discussion of the results obtained in detail. Lastly, Section 5 of the paper closes the paper by providing summative information of the major findings of the research work and providing possible future research directions.

2. Related works

Over the last few years, deep learning and metaheuristic optimization methods have been actively used in the field of overcoming the escalating menace of the Distributed Denial-of-Service (DDoS) attacks in the cloud, Internet of Things (IoT), and edge computing systems. In general, the literature overview shows that although deep learning and metaheuristic optimization are important in DDoS detection, most of the solutions have limitations like suboptimal hyperparameter optimization, premature convergence, or excessive computation. Furthermore, few studies have been able to apply opposition-based learning techniques to LSTM models to detect DDoS in the cloud environment. Such constraints serve as the driving force behind the suggested study that presents an optimized LSTM model with an IFA-LSTM to attain a better detection rate, a faster rate of convergence, and a better tolerance to cloud computing conditions.

Table 1: Comparison of recent related papers

Ref. No	Author (Year)	Method	Dataset	Merits	Limitations
[1]	Gupta et al. (2023)	Edge-cCNN + Cuckoo Search	IoT traffic dataset	Lightweight, high accuracy, edge-friendly	No temporal modelling, limited cloud validation
[2]	Subramanian et al. (2022)	NDCS (Multi-objective Cuckoo Search)	Google Cluster	Secure VM migration, reduced energy, and makespan	Not focused on DDoS detection
[3]	Hu et al. (2024)	Survey (Nature-inspired IDS)	Multiple datasets	Comprehensive review, identifies trends	No experimental results
[4]	Preethi et al. (2023)	CS-GWO + IVM	Benchmark IDS datasets	Effective feature selection handles imbalance	Static classifier, limited temporal learning
[5]	Parkash et al. (2022)	Cuckoo Search (New fitness)	KDD Cup 99	Improved optimization efficiency	Uses an outdated dataset
[6]	Senthilkumar et al. (2025)	IRAEN + MCSO	Network traffic dataset	Attention-based feature extraction	High computational complexity
[7]	Abed et al. (2023)	Cuckoo Search-based routing	SDN environment	Reduced latency and routing cost	Not an IDS model
[8]	Ganne (2023)	AI/ML-based IDS framework	CIADA, Packet	Predictive security framework	No optimization validation
[9]	Sumathi et al. (2024)	ANN + GWO + SOM	UNSW-NB15	Low false alarms, fast prediction	ANN is weak in long-term dependencies
[10]	Hashemi et al. (2022)	Multi-objective GWO	Fog scenarios	Reduced energy & SLA violations	No attack classification
[11]	Nkongolo et al. (2022)	Ensemble ML + GA	UGRansome1819	Strong zero-day detection	High computational cost

[12]	Sughanthini et al. (2024)	POEHO-LSTM	NSL-KDD, ISCXIDS-2012	Captures temporal patterns effectively	Convergence overhead in large data
[13]	Al-Khayyat et al. (2024)	HAFSO-DSAE	Android APKs	High malware detection accuracy	Not DDoS-focused
[14]	Bakro et al. (2024)	GOA-GA + RF	UNSW-NB15, CIC-DDoS2019	Handles imbalance & redundancy	RF ignores sequential behavior
[15]	Sumathi et al. (2023)	HHO-PSO + BPN/MLP	NSL-KDD	Improved parameter tuning	Limited scalability
[16]	Jain et al. (2023)	PSO & Firefly	Cloud environment	Improved threat analysis	No DL-based IDS
[17]	Benni et al. (2024)	PSO & ACO	Network traffic	Optimized routing & mitigation	Not a full IDS
[18]	Shrivastava et al. (2023)	Ensemble FS + DT	NSL-KDD	Significant feature reduction	Overfitting risk
[19]	Reddy et al. (2024)	GA-AOA + Ensemble Voting	Cloud traffic	High mitigation rate	Complex architecture
[20]	Srilatha et al. (2022)	PSO + DNN	CICIDS2017	Very high accuracy	High training cost
[21]	Arunadevi et al. (2022)	APO-BPNN	IDS benchmarks	Faster convergence	Weak temporal modelling
[22]	Ali et al. (2024)	PSO-ML Hybrid	UNSW-NB15	High accuracy across datasets	Increased processing time
[23]	Naiem et al. (2022)	Survey on DDoS defense	—	Identifies research gaps	No implementation

Based on the detailed overview of the recent literature on the topic of DDoS attack detection in cloud, IoT, and edge computing platforms, it is possible to define several research gaps that are particularly critical. Despite the high detection rate of most of the established methods based on deep learning and machine learning models, a number of them utilize fixed type classifiers or shallow neural networks, which cannot be effective in identifying long term temporal structures of network traffic data. This consequently leads to poor performance in responding to the changing and low-rate patterns of DDoS attacks.

A number of studies use metaheuristic optimization schemes like PSO, GA, GWO, Cuckoo Search, and their hybrids to select features or perform parameter optimization. But the majority of these algorithms have poor premature convergence and exploration behavior, particularly when dealing with high-dimensional cloud traffic data. As a result, the optimization models can end up with suboptimal solutions, which bring about instability and poor generalization. Though some of the studies unify opposition-based learning with optimization methods, its application is still scarce and is usually confined to full opposition strategies, which can come at a higher computational cost. The possibility of partial opposition-based learning in terms of population diversity increase, convergence acceleration, and optimization efficiency improvement has not been adequately studied in the light of LSTM-based DDoS detection in the cloud environment. Moreover, much of the current methods is aimed at obtaining large accuracy with little concern on the convergence behavior, the computer complexity and scalability, which are very vital conditions in real-time cloud intrusion detection systems. Moreover, the absence of a standard assessment among benchmark datasets includes UNSW-NB15 and CICIDS2017, means that it is hard to compare fairly and practice the deployment of the research. In order to overcome these drawbacks, it is evident that a more efficient, time-conscious, and optimization-based deep learning model is required that can be capable of high detection accuracy and produce quicker convergence, robustness, and scalability. This study is expected to fill this gap by suggesting an optimized LSTM-based DDoS detection model using Improved Firefly Alcohol with Partial Opposition-Based Learning that would be specifically provided within cloud computing environments.

3. Research Methods

The following subsections are discussed about the research methods which are used in this research work.

3.1 Autoencoder

Autoencoders (AEs) are made up of a decoder and an encoder, which were trained to minimize the reconstruction error as they reproduce their input. As the decoder uses the learned features in a bid to make an attempt to replicate the same input, the encoder extracts the salient features of the input vector. Considering an input matrix $x_i^{m\times n}$ with m samples, and n. The dimensionality of the learned feature space (the number of hidden units) may be denoted as l, which is less than n, and thus the number of hidden layers between the encoder and decoder levels may be more than one. In the case of using more than one of these layers, the construct is known as a deep autoencoder. The encoder compares the input vector x i through a nonlinear mapping to a hidden representation h_j $(j = 1, \ldots, l)$ denoted by.

$$h_j = \sigma(\sum_{i=1}^{n} W_{ij} \times x_i + b_j) \quad (1)$$

Where σ is the nonlinear activation function. W_{ij} is the weight and b_j is the bias term. The decoder then maps the hidden representation to its original representation. Reconstruction of the image of the ith element is as follows:

$$\widetilde{x_i} = \sigma(\sum_{j=1}^{l} \widehat{W}_{ij} \times h_j + \hat{b}_j \quad (2)$$

The average reconstruction error (MSE) of the original and reconstructed input is used to optimize the hyperparameters of the autoencoder and reduce the reconstruction loss between x and the reconstruction, and the loss functional is given as,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\|x_i - \widetilde{x_i}\|^2 \quad (3)$$

3.2 Long short-term memory (LSTM)

Long-term dependencies can be learnt in LSTM, a special RNN architecture invented by Hochreiter and Schmidhuber [24]. As indicated, various gates that control the cell at any given point in time t may maintain the value or reconfigure it based on the state of the gates. Three gates (i.e. forget gate (f_t), input gate (i_t), and output gate(o_t)) are applied to the cell. Also, there is a candidate value entrance modulation gate. The gates may be described as follows:

$$i_t = \sigma(W_{x,i}x_t + W_{i,h}h_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(W_{f,i}x_t + W_{f,h}h_{t-1} + b_f) \quad (2)$$
$$o_t = \sigma(W_{o,i}x_t + W_{o,h}h_{t-1} + b_o) \quad (3)$$
$$c'_t = tanh(W_{c',i}x_t + W_{c',h}h_{t-1} + b_{c'}) \quad (4)$$

In this case, W synaptic weight matrix, x_t- actual input, b - the bias vectors, the vector c t is fresh candidates that might be inserted into the current state of the cell. h_{t-1}- is the previous output of the LSTM at time t-1. The activation functions that are analogous to sigmoid and tangent hyperbolic activation functions are s$\sigma(.)$and $tanh()$. The first step of the LSTM algorithm is to choose the percentage of the previous memory rate that will be deducted from the state of the cell. This is the decision of the forget gate. The input gate will decide the extent of the fresh information to be stored in the next stage. Subsequently, the state of the cell can be determined by using the following phrase:

$$c_t = f_t \odot c_{t-1} + i_t \odot c'_t \quad (5)$$

Where, $\odot$ is the elementwise product, h_t the defined LSTM output height can be defined as follows,

$$h_t = o_t \odot tanh(c_t) \quad (6)$$

The disappearance of gradient is a problem with traditional RNNs. In particular, the slopes of the loss function tend to zero, the number of layers with the same activation function is used, and it becomes difficult to teach the network with the help of backpropagation of errors. The LSTM uses memory cells, in which each cell stores a cell state and a hidden cell state, and applies three gates (to be specific, input gate, output gate, and forget gate) to control the information flowing into or out of the memory cell to eliminate the vanishing gradient problem.

3.3 Firefly algorithm

The change in the intensity of light and the setting up of attractiveness is what is actually the main concept of the firefly optimization algorithm [25]. The objective function is associated with brightness, and we consider it to determine the attractiveness of a firefly. Suppose, then, that we have a swarm of fireflies with each x_i the possible solution of a given firefly i. The brightness is optimally selected to demonstrate that this current position (x) has a fitness value $f(x_i)$

$$I_i = f(x_i), 1 \le i \le n \quad (8)$$

The attractiveness is determined by the intensity of light that the surrounding fireflies observe. Each firefly possesses its attraction parameter, which is represented by the value of β This value defines the ability of the firefly to effectively attract other members of the swarm. This appeal varies depending on the distance between fireflies i and j and in positions a and b, which can be stated as follows.

$$r_{ij} = \|x_i - x_j\| \quad (9)$$

The firefly's attractiveness was considered below

$$\beta(r) = \beta_0 e^{-\gamma r^2} \quad (10)$$

Where, β_0 is the reflection of attractiveness. γ is the Light absorption coefficient. A firefly position is attracted to an alternative expressed by

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}(x_i - x_j) \quad (11)$$

3.3 Partial opposition-based learning (POBL)

Z. Hu et al. (2014) established POBL [26] and an opposite point $\begin{bmatrix} x_1 & \acute{x}_2 & \acute{x}_3 \\ \acute{x}_1 & x_2 & \acute{x}_3 \end{bmatrix}$ has opposite values of the original. The partial opposing points of a given point X can be described as follows

$$p\acute{\chi}1 = \begin{bmatrix} p\acute{\chi}_1^1 \\ p\acute{\chi}_2^1 \\ \cdots \\ p\acute{\chi}_d^1 \end{bmatrix}_{D\times 1} = \begin{bmatrix} x_1 & \acute{x}_2 & \acute{x}_3 & \dots & \acute{x}_D \\ \acute{x}_1 & x_2 & \acute{x}_3 & \dots & \acute{x}_D \\ \vdots & \vdots & \vdots & \ddots & \dots \\ \acute{x}_1 & \acute{x}_2 & \acute{x}_3 & \dots & x_D \end{bmatrix}_{D\times D} \quad (11)$$

The position of each firefly is initialized as follows,

$$X_j^1(t)|_{(t=0)} = X_j^{min} + \left(X_j^{max} - X_j^{min}\right).r_{ij}^u(t)|_{(t=0)} \quad (12)$$

$r_{ij}^u(t)|_{(t=0)}$ is a randomly uniformly dispersed number with a range between 0 and 1. The random and its opposite positions are chosen based on their fitness for the best initial position of the firefly. The opposite location of firefly is updated in the search space as follows:

$$\acute{x_{ij}}(t) = a_j(t) + b_j(t) - \alpha_{ij}(t).x_{ij}(t) \quad (13)$$

Where $[a_j(t), b_j(t)]$ is the dynamic search space with a range of 0 and 1.

3.4 POBL-based FA

In fact, POBL is a fairly clever trick, which enables the Firefly Algorithm to overcome some of its major flaws, such as overconvergence and lack of diversity. In the traditional FA, we have fireflies only in pursuit of the more brilliant ones, and this promotes much exploitation, though at the price, generally, of exploration of new regions. POBL gets around this by generating half-opposite solutions and providing the swarm with new diversity without discarding the good solutions they already discovered. Rather than flipping all the dimensions as is the case with full opposition-based learning, POBL flips a selected part of the coordinates. In this manner, we get the benefit of the valuable information in the untouched dimensions, and we also get to explore additional areas of the search space. A more controlled exploration is the consequence of this, which makes the algorithm more powerful when dealing with complex, high-dimensional problems.

The other major advantage of POBL is that it accelerates convergence. Our chances of reaching a point nearer to the global optimum are higher by considering both the original positions and their partial opposites, and, in particular, in the initial stages. The dual-view system reduces unwarranted movements and increases efficiency. POBL is also used to get FA out of local traps. As the attracting nature of certain positions of the fireflies may cause them to linger in a local optimum, adding partial opposites allows the algorithm to jump to new places that it has not explored before, or at least, visited less frequently. That is highly important when multimodal optimization is required, and there are multiple local peaks. An incorporation of POBL into FA helps to make the approach stronger and more stable. This results in higher-quality solutions being pulled off by the POBL-improved FA since it better balances exploration and exploitation, and is also known to converge more quickly and produce consistent results across successive runs.

3.5 Proposed optimized LSTM

The hyperparameter optimization of the LSTM model with (FA-POBL) is a minimization problem with Mean Squared Error (MSE) as the fitness function. The fireflies are candidates of the LSTM hyperparameter vectors in this framework, and they contain the following parameters: the number of hidden neurons, the number of LSTM layers, learning rate, batch size, dropout rate, and the number of epochs. Each hyperparameter has appropriate lower and upper limits that are used to make a search space that is feasible and well-constrained. Once the process of initializing the fireflies is completed, the intensity of each firefly can be evaluated by training the LSTM model using the specified hyperparameters and computing the MSE on the validation set. MSE being the mean of the squared difference between the predicted and the actual value, the lower the MSE, the higher the accuracy of the prediction, hence the brighter. MSE is particularly appropriate as a guide to metaheuristic optimization when applied to regression and time series prediction problems. N The fireflies that have greater MSE values in the optimization process are attracted to bright fireflies with lower MSE values based on the attraction rule of the Firefly Algorithm. The movement maximizes the hyperparameter settings that have promising values and supports the exploration of high-quality regions in the search space. Concurrently, a randomization parameter helps in making sure that exploration is sufficiently large, so that the algorithm is not prematurely drawn to suboptimal LSTM hyperparameters. The definition of the objective function is as follows:

$$MSE = \frac{1}{N}(y_i - \hat{y}_i)^2 \quad (23)$$

where y_i is the forecast value and $\hat{y}_i$ is the actual value. POBL is further added to improve convergence and avoid local minima. Rather than generating an entirely different hyperparameter array, POBL only opposes a few dimensions, including the number of hidden units or the learning rate. The original and partial opposite solution of the firefly is tested with the MSE criterion, and a configuration with the lowest MSE is retained. The approach increases the chance of finding improved hyperparameter settings without wiping out useful parameter values. The process of optimizing FA-POBL is repeated until a stopping criterion is reached (e.g. sufficient generations or insignificant reduction in MSE. The resultant output is the hyperparameter setting that leads to the smallest validation MSE. The POBL-enhanced FA enables the LSTM model to produce a higher accuracy in prediction, a quicker convergence, and more consistent generalization behavior, which makes it very effective in prediction problems that are nonlinear and complex.

4. Experimental results and analysis

To check the effectiveness and reliability of the proposed model, the analysis and experimental results are necessary. They offer objective data on performance gains made with the help of autoencoder-based feature selection and FA-based LSTM, based on measuring accuracy, precision, recall, F-score, and convergence behaviour. The analysis can be made more detailed to have meaningful comparison with baseline and current methods, learn stability and optimization efficiency, as well as strengths and limitations of the approach. All in all, the practical applicability, robustness, and scalability of the suggested intrusion detection framework in real-world network settings are supported by the experimental findings and the analysis. The MATLAB2022R was used to implement the detection method.

4.1 Datasets details

UNSW-NB15 dataset [27] *is* subset of 257,673 examples and 2,540,044 examples of 48 features are used. The sample sizes of the training (175,341) and testing (82,332) sets are 175,341 and 82,332, respectively. There are nine types of attacks, namely: worms, backdoors, exploits, fuzzers, shell-code, and DDoS. CIC-IDS2017 dataset [28] are five days of traffic between Monday and Friday, and it is detailed. During the other days, there are attacks and usual exist, but only normal samples on Monday. The data is of eight different types of attacks: Botnet, Bruteforce, DDoS, DoS, Heartbleed, Infiltration, Portscan, and Web. The number of examples is 2,491,689, the average is 2,273,097, and 218,592 are attacked with 78 structures.

Table 2 : Feature importance score for UNSW NB-15 dataset using Auto Encoder

Feature Name	Importance score	Rank
dload	0.000000	0
spkts	0.151811	1
ct_src_dport_ltm	0.269702	2
ct_dst_ltm	0.346930	3
sinpkt	0.480108	4
dwin	0.518893	5
sload	0.526516	6
tcprtt	0.557248	7
smean	0.567718	8
dttl	0.586537	9
trans_depth	0.594095	10
is_ftp_login	0.618666	11
dur	0.627868	12
dtcpb	0.638538	13
sttl	0.684201	14
dpkts	0.691070	15
sjit	0.706511	16
ct_dst_src_ltm	0.713066	17
ct_srv_src	0.727549	18
service	0.738994	19
ackdat	0.745012	20
djit	0.817505	21
dmean	0.826080	22
swin	0.826722	23
ct_state_ttl	0.845147	24
state	0.866092	25
dloss	0.901294	26
sloss	0.902183	27
rate	0.903689	28
proto	0.909601	29
dbytes	0.928522	30
stcpb	0.932013	31
ct_ftp_cmd	0.934829	32
sbytes	0.935932	33
response_body_len	0.949827	34
synack	0.958242	35
dinpkt	0.987970	36
ct_dst_sport_ltm	1.000000	37

4.2 Data preprocessing

Raw network traffic data is then cleansed by deleting duplicate records, treating missing values, and deleting irrelevant and inconsistent records. Appropriate encoding of categorical attributes is done to represent them in a numerical form, and numerical features are normalized or standardized so that they have equal scale. Rapid resampling methods are used in order to deal with the imbalance of classes. Lastly, the processed data are divided into training and testing data and run through an autoencoder to discover small representations, which allow the selection of the best features at the expense of classification. Data is normalized by the Min- Max [29, 30] scaling method to bring all the numerical variables to a common range [0,1]. This is done to make sure that features that have higher numeric values do not take over during the learning process and enhance the convergence of the LSTM model. The original data distribution is not lost with this normalization, which allows the optimization-based IFA-LSTM model to train faster and achieve better results.

$$z = \frac{x - min(x)}{max(x) - min(x)} \quad (24)$$

Where, x is the present value, and $max(x)$ and $min(x)$ are its upper and lower bounds. The trained example is represented by the composed dataset (70%), and the testing example is represented by the other 30 %. Tables 2 and 3 (Figures 1 and 2) discuss the feature importance score of UNSWNB-15 and CICIDS-2017 datasets, respectively.

4.3 Parameters settings

The hyperparameter optimization is conducted to increase the learning ability and the generalization performance of the LSTM model. This paper uses the IFA to auto-tune very important LSTM hyperparameters that include learning rate, hidden neurons, batch size, dropout rates, and epochs. IFA can intelligently balance exploration and exploitation and therefore is able to efficiently search the best parameter space, prevent local optima, and speed up the convergence. This optimization procedure leads to better accuracy and stability and less time to train than manually adjusted and traditional optimization-based LSTM models.

The effectiveness of hyperparameter optimization is dependent on the parameters of the Firefly Algorithm (FA).

Table 3: Feature importance for CICIDS-2017 dataset using Auto Encoder

Feature Name	Importance score	Rank	Feature Name	Importance score	Rank
Bwd PSH Flags	0	0	Bwd Packet Length Min	0.652004	39
Fwd Avg Bytes/Bulk	0	1	Down/Up Ratio	0.653023	40
Fwd Avg Packets/Bulk	0	2	min_seg_size_forward	0.677089	41
Bwd Avg Bytes/Bulk	0	3	SYN Flag Count	0.677513	42
CWE Flag Count	0	4	Fwd PSH Flags	0.678352	43
Bwd Avg Packets/Bulk	0	5	Active Max	0.75508	44
Bwd Avg Bulk Rate	0	6	Fwd Packet Length Mean	0.760739	45
Bwd URG Flags	0	7	Avg Fwd Segment Size	0.760746	46
Fwd URG Flags	0	8	Fwd Packet Length Max	0.764629	47
Fwd Avg Bulk Rate	0	9	Fwd Packet Length Std	0.771227	48
Bwd IAT Total	0.411608	10	Active Mean	0.771499	49
PSH Flag Count	0.421377	11	Bwd IAT Min	0.782112	50
Bwd IAT Max	0.424993	12	Fwd IAT Min	0.810281	51
Bwd IAT Std	0.441984	13	Active Min	0.818087	52
FIN Flag Count	0.475182	14	Active Std	0.83311	53
Fwd IAT Std	0.48549	15	URG Flag Count	0.93475	54
Idle Mean	0.490261	16	Total Length of Fwd Packets	0.94501	55
Idle Max	0.491562	17	Subflow Fwd Bytes	0.945324	56
Fwd IAT Max	0.492962	18	Bwd Header Length	0.985889	57
Flow IAT Max	0.493276	19	Fwd Header Length	0.985928	58
Flow IAT Std	0.508462	20	Fwd Header Length.1	0.985949	59
Bwd Packet Length Mean	0.517305	21	Subflow Bwd Packets	0.985952	60
Avg Bwd Segment Size	0.517306	22	Total Backward Packets	0.985968	61
Init_Win_bytes_forward	0.527807	23	Subflow Fwd Packets	0.986026	62
Fwd IAT Total	0.535289	24	Total Fwd Packets	0.986032	63
Packet Length Std	0.53544	25	Total Length of Bwd Packets	0.986109	64
Flow Duration	0.535892	26	Subflow Bwd Bytes	0.986171	65
Packet Length Mean	0.540261	27	act_data_pkt_fwd	0.986222	66
Average Packet Size	0.547044	28	Idle Std	0.996599	67
Flow Packets/s	0.558854	29	Fwd Packet Length Min	1	68
Bwd Packet Length Max	0.56605	30	Min Packet Length	1	69
Bwd Packet Length Std	0.574081	31	Bwd Packets/s	1	70
Max Packet Length	0.583778	32	Flow IAT Min	1	71
ACK Flag Count	0.587196	33	Fwd Packets/s	1	72
Bwd IAT Mean	0.616234	34	ECE Flag Count	1	73
Flow IAT Mean	0.616721	35	Init_Win_bytes_backward	1	74
Fwd IAT Mean	0.629436	36	Flow Bytes/s	1	75
Packet Length Variance	0.630707	37	RST Flag Count	1	76
Destination Port	0.633958	38			

Direct impact on search diversity is the population size, in which a larger size enhances global exploration, but it adds to the computational load. The convergence speed is determined by the first attractiveness, which determines the magnitude to which fireflies approach superior solutions. Light absorption coefficient (γ) balances both global and local search, with low γ values promoting broader searching, whereas higher values promote finer exploitation around promising areas. Practically, the randomization parameter (α) aids the algorithm to get out of local optima as the algorithm creates stochastic movements, especially during initial iterations. The combination of these two parameters allows faster convergence and better quality of solutions and ensures stable optimization and, as a result, better performance of the IFA-LSTM intrusion detection model. Table 4 shows the parameter settings of the FA and LSTM methods.

4.4 Performance measures

Performance analyzers are used to assess the ability of ML. The accuracy, precision, recall, and f-measures are four pointers used as follows,

- The accuracy of classification is measured by the proportion of correctly identified samples and the total number of samples.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (22)$$

- Precision is defined as the ratio of accurately categorized samples to projected positive samples.

$$Precision = \frac{TP}{FP+TP} \quad (23)$$

- The Recall is defined as the ratio of probable positives to the sum of genuine positives and false negatives.

$$Recall = \frac{TP}{TP+FN} \quad (24)$$

- F-measures represent the harmonic mean of recall and precision as follows

$$F - Measures = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (25)$$

False positive (FP) means a mistakenly expected regular, whereas false negative (FN) indicates an incorrectly predicted

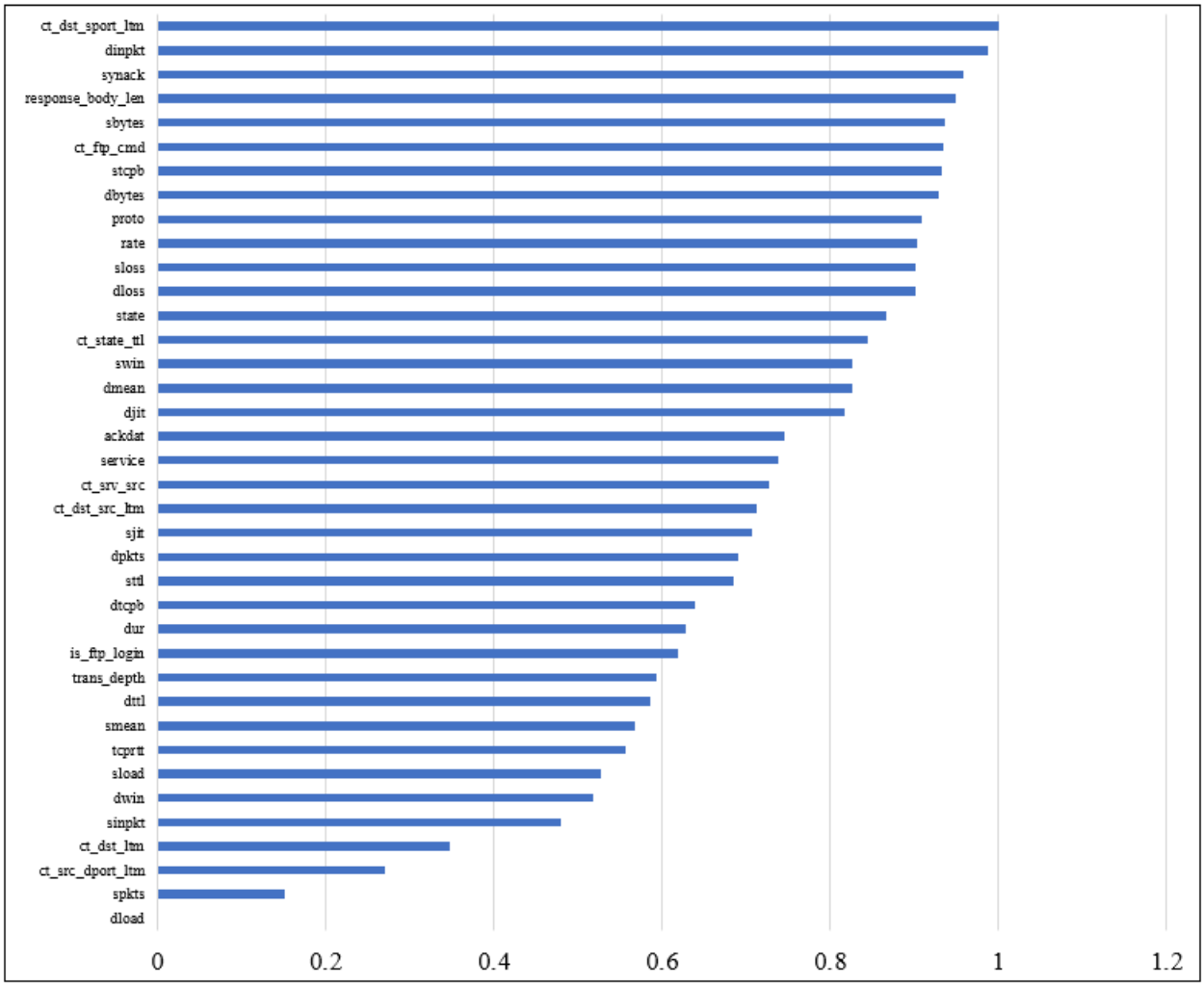

Figure 1: Feature importance score for UNSW NB-15 dataset using Auto Encoder

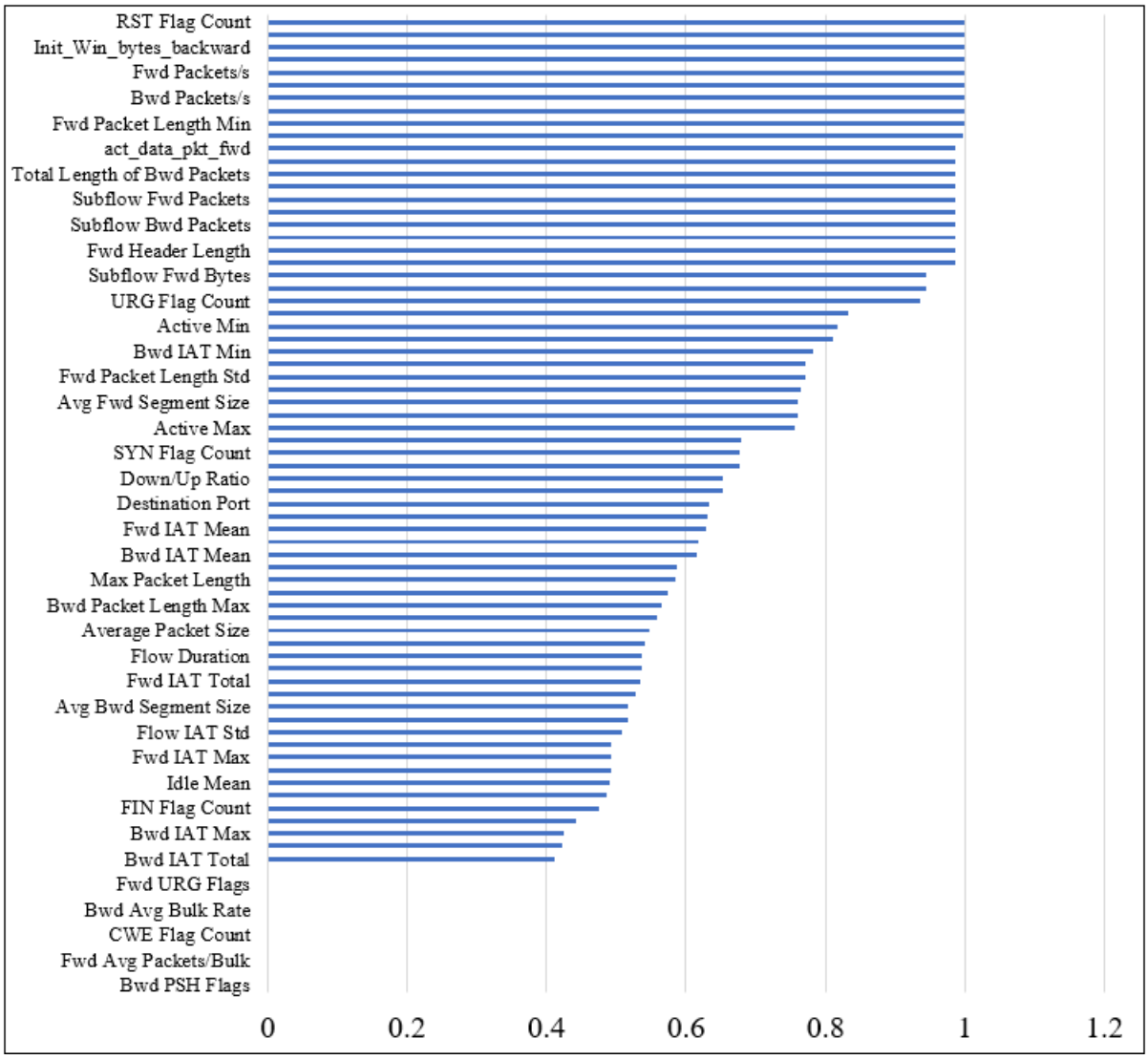

Figure 2: Feature importance score for CICIDS2017 dataset using AE

DDoS attack. True positive (TP) indicates a successfully predicted DDoS attack, whereas true negative (TN) indicates a properly recognized normal.

4.5 Results analysis

The analysis of the results is important in supporting the effectiveness of the proposed model through the systematic comparison of its performance with the base and current methods based on the conventional evaluation mechanisms. It aids the illustration of the effect of the hyperparameter optimization and the autoencoder-driven selection of features on the accuracy, the stability, and the convergence behavior. The results analysis is the empirical evidence of the robustness, generalization capability, and the efficiency of the computations of the implementation through detailed analysis of the performance measures and convergence trends, which justifies the appropriateness of the suggested approach to real-life applications of intrusion detection.

Table 5 and Figure 3 show the performance and convergence analysis of the UNSW-NB15 dataset without feature selection. In Table 5, it can be seen that all the optimization-based LSTM models are significantly better than the baseline LSTM in terms of accuracy, recall, precision, and F-score. The proposed IFA-LSTM has the best performance in all measures, which proves that it has a great ability to learn the complicated intrusion patterns. This finding is also supported by Figure 3, which demonstrates that IFA-LSTM reaches an optimal solution with fewer iterations and stabilizes, showing that it is an effective way of learning and prevents the existence of local optima. The outcomes of the CICIDS2017 dataset before feature selection are explained in Table 6 and Figure 4. As indicated in Table 6, IFA-LSTM once again achieves the highest accuracy and F-score with respect to FA-LSTM, IPSO-LSTM, PSO-LSTM, GA-LSTM, and the conventional LSTM model. This dataset has better improvement as it is highly dimensional and diverse in traffic. The convergence curve of the IFA-LSTM model achieves the lowest fitness value sooner than the rest of the models, as shown in Figure 4, which indicates that it converges more quickly and is more stable at training. Table 7 and Figure 5 show the effects of the feature selection on the UNSW-NB15 data post-selection. In Table 7, a slight decrease in the absolute accuracy is observed in all the methods, but even in all the evaluation metrics, the proposed IFA-LSTM remains in higher performance.

Table 4: Parameter Settings

LSTM				FA	
Parameters	Values	Parameters	Values	Parameters	Values
Activation	TanH, Sigmoid	Expected error	0.0005	N	50
Loss function	MSE	Weight range	-0.5 and 0.5	β_0	1
Learning rate	0.005	Number of hidden neurons	32	λ	1
Epochs	500	Dropout	0.1	T	100

Table 5: Performance analysis for the UNSW-NB15 dataset before feature selection

Methods	Accuracy (%)	Recall (%)	Precision (%)	F-Score (%)
IFA-LSTM	**98.92**	**98.75**	**99.71**	**99.23**
FA-LSTM	97.68	97.12	98.36	97.73
IPSO-LSTM	97.31	95.48	97.89	96.67
PSO-LSTM	95.86	94.21	96.44	95.31
GA-LSTM	93.72	93.10	95.68	94.37
LSTM	91.94	92.88	94.36	93.61

Table 6: Performance analysis for the CICIDS2017 dataset before feature selection

Methods	Accuracy (%)	Recall (%)	Precision (%)	F-Score (%)
IFA-LSTM	**99.12**	**99.04**	**99.36**	**99.20**
FA-LSTM	98.41	98.02	98.67	98.34
IPSO-LSTM	97.63	96.88	97.94	97.40
PSO-LSTM	96.42	95.76	96.88	96.31
GA-LSTM	95.18	94.69	95.92	95.30
LSTM	93.87	94.12	93.44	93.78

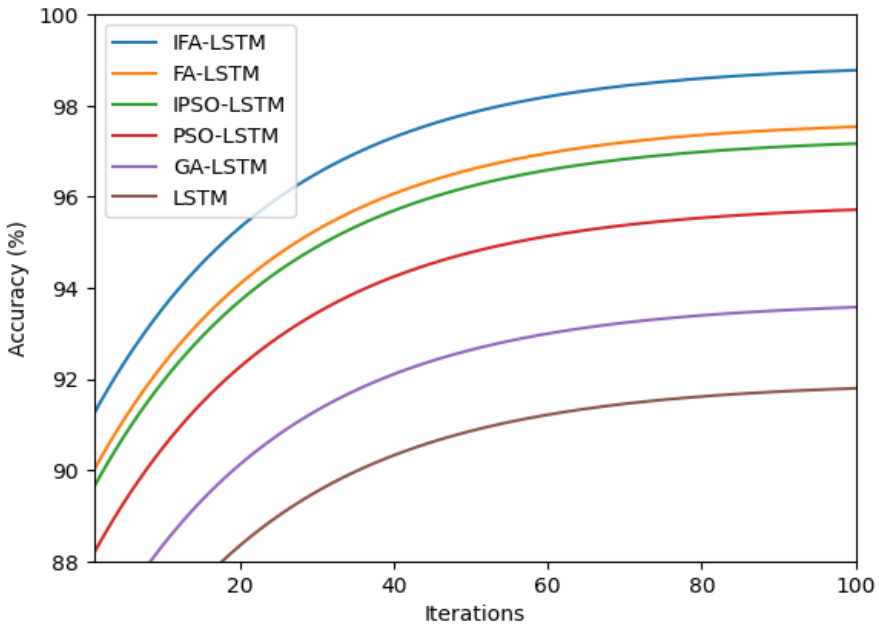

Figure 3: Convergence analysis for UNSW-NB15 dataset before feature selection

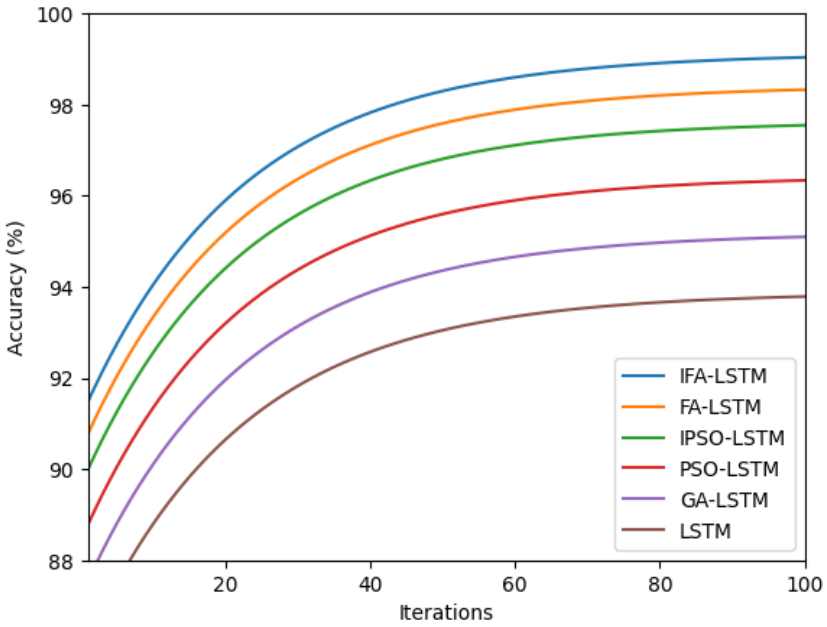

Figure 4: Convergence analysis for the CICIDS2017 dataset before feature selection

Table 7: Performance analysis for UNSW-NB15 dataset after feature selection

Methods	Accuracy (%)	Recall (%)	Precision (%)	F-Score (%)
IFA-LSTM	**97.84**	**97.62**	**98.55**	**98.12**
FA-LSTM	96.41	95.88	97.14	96.59
IPSO-LSTM	95.92	94.37	96.48	95.52
PSO-LSTM	94.28	92.81	95.21	94.17
GA-LSTM	92.36	91.94	94.12	92.98
LSTM	90.51	91.62	92.88	91.46

Table 8: Performance analysis for the CICIDS2017 dataset after feature selection

Methods	Accuracy (%)	Recall (%)	Precision (%)	F-Score (%)
IFA-LSTM	**97.96**	**97.84**	**98.21**	**98.02**
FA-LSTM	96.88	96.41	97.12	96.73
IPSO-LSTM	95.94	95.21	96.63	95.87
PSO-LSTM	94.73	94.08	95.44	94.69
GA-LSTM	93.36	92.84	94.21	93.51
LSTM	91.88	92.14	91.63	91.92

Figure 4 demonstrates the convergence pattern to be smoother and faster than the pre-feature selection case, which reveals the less complexity of computation and enhanced stability because unneeded features are dropped. Lastly, Table 8 and Figure 6 indicate the results of various models on the CICIDS2017 dataset following feature selection. As it is observed in Table 7, IFA-LSTM has the highest accuracy, the highest recall, the highest precision, and the highest F-score, which proves the strength despite the reduced set of features. Figure 6 provides better convergence, smoothness, and stability of all models, with the convergence of IFA-LSTM longest. All in all, the findings in Tables 5-8 and Figures 3-6 strongly confirm the fact that the suggested IFA-LSTM architecture, with feature selection, is an accurate, stable, and computationally efficient method of intrusion detection in large-scale network settings.

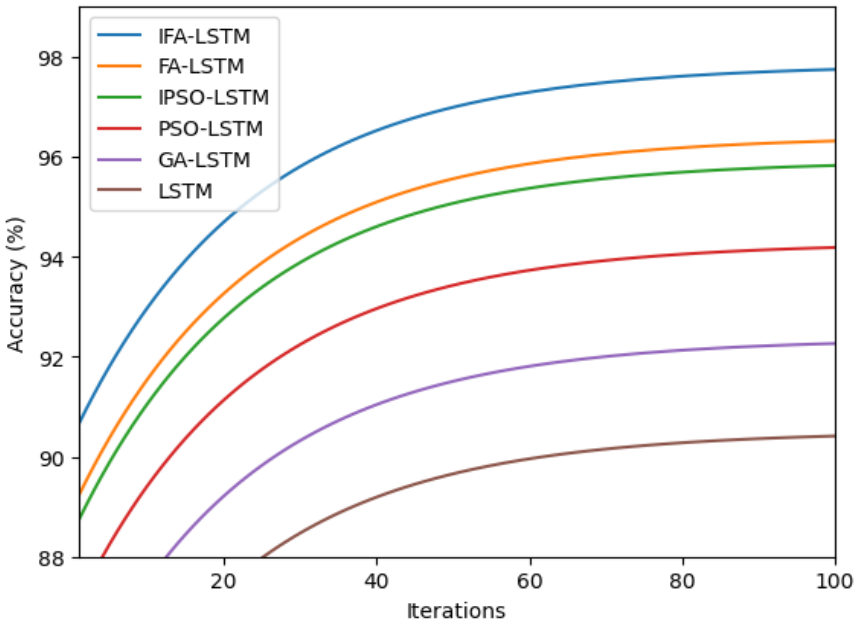

Figure 5: Convergence analysis for UNSW-NB15 dataset after feature selection

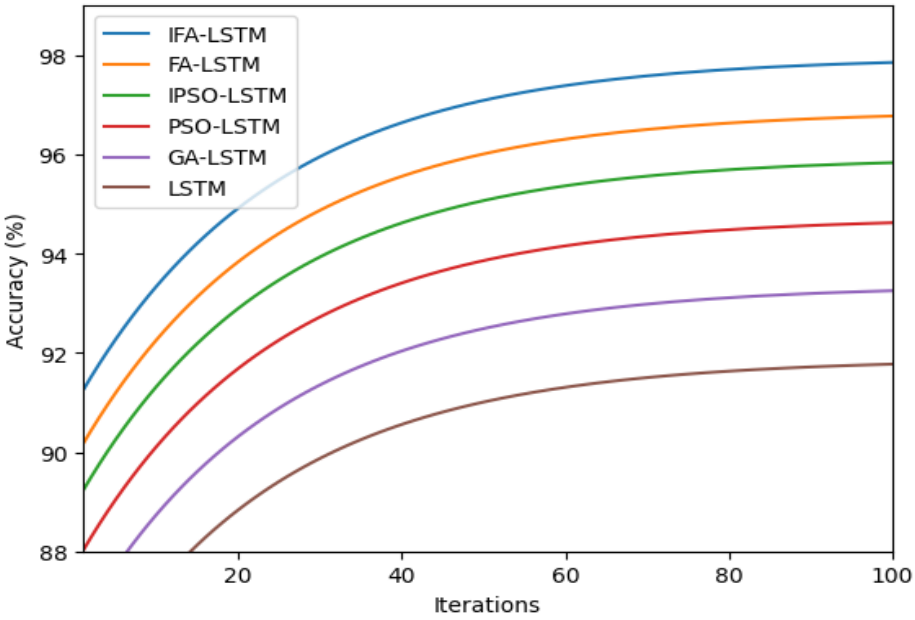

Figure 6: Convergence analysis for the CICIDS2017 dataset after feature selection

5. Conclusions

This paper introduced an IFA-LSTM model optimized with IFA in conjunction with an optimal choice of features using AE to detect intrusion in an efficient way. The autoencoder was able to acquire compact and informative latent representations on high-dimensional network traffic data, thus allowing the elimination of redundant and irrelevant features, and critical intrusion characteristics were retained. Preliminary experimental analyses on the UNSW-NB15 and CICIDS2017 datasets showed that the combination of autoencoder-based feature selection methods had significant benefits in terms of decreasing the complexity of computations and enhancing the stability of convergence with no compromise on detection accuracy. IFA-LSTM was also more accurate, precise, and had a higher recall and F-score than the baseline LSTM and other variants of the same model that were optimized in accuracy and speed, with or without feature selection. In general, the combination of the auto-encoders-based optimal feature selection algorithm and intelligent metaheuristic-based LSTM optimization algorithm led to the development of a powerful, precise, and scalable intrusion detection system applicable to large-scale and high-dimensional network configurations. This pattern can further be expanded to future work with hybrid strategies of feature selection, attention, and real-time implementation of this methodology in dynamic network contexts.

References

[1] B. B. Gupta, A. Gaurav, K. T. Chui, and V. Arya, "Optimized edge-cccn based model for the detection of ddos attack in iot environment," in International Conference on Edge Computing, 2023: Springer, pp. 14-23.

[2] N. Venkata Subramanian and V. Shankar Sriram, "An effective secured dynamic network-aware multi-objective cuckoo search optimization for live VM migration in sustainable data centers," Sustainability, vol. 14, no. 20, p. 13670, 2022.
[3] W. Hu, Q. Cao, M. Darbandi, and N. Jafari Navimipour, "A deep analysis of nature-inspired and meta-heuristic algorithms for designing intrusion detection systems in cloud/edge and IoT: state-of-the-art techniques, challenges, and future directions," Cluster Computing, vol. 27, no. 7, pp. 8789-8815, 2024.
[4] P. Preethi, I. Vasudevan, S. Saravanan, R. K. Prakash, and A. Devendhiran, "Leveraging network vulnerability detection using improved import vector machine and Cuckoo search based Grey Wolf Optimizer," in 2023 1st International Conference on Optimization Techniques for Learning (ICOTL), 2023: IEEE, pp. 1-7.
[5] D. Parkash and S. Mittal, "An Enhanced Secure Framework Using CSA for Cloud Computing Environments," in International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Volume 2, 2022: Springer, pp. 349-356.
[6] A. Senthilkumar, L. Kathirvelkumaran, K. Kavitha, N. Sampathkumar, and S. Sureshkumar, "Develop a Hybrid Improved Residual Attention with Efficient Net with Mosquito-Cuckoo Search Optimization to Detect the Attack during Network Traffic."
[7] S. S. Abed and M. N. Fadhil, "Optimization Flow Control In Software-Defined Networking Using Cuckoo Search Algorithm," Journal of Namibian Studies, vol. 33, 2023.
[8] A. Ganne, "IoT threats & implementation of AI/ML to address emerging cyber security issues in IoT with cloud computing," International Research Journal of Modernization in Engineering Technology and Science, vol. 5, pp. 1-5, 2023.
[9] S. Sumathi and R. Rajesh, "HybGBS: A hybrid neural network and grey Wolf optimizer for intrusion detection in a cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 36, no. 24, p. e8264, 2024.
[10] S. M. Hashemi, A. Sahafi, A. M. Rahmani, and M. Bohlouli, "Gwo-sa: Gray wolf optimization algorithm for service activation management in fog computing," IEEE Access, vol. 10, pp. 107846-107863, 2022.
[11] M. Nkongolo, J. P. Van Deventer, S. M. Kasongo, S. R. Zahra, and J. Kipongo, "A cloud based optimization method for zero-day threats detection using genetic algorithm and ensemble learning," Electronics, vol. 11, no. 11, p. 1749, 2022.
[12] V. Sughanthini and P. Bharathisindhu, "DDoS Attack Detection Using Optimized Long Short-Term Based on Partial Opposition-Based Swarm Intelligence Algorithm," in 2024 2nd International Conference on Sustainable Computing and Smart Systems (ICSCSS), 2024: IEEE, pp. 532-537.
[13] A. Al-Khayyat, M. A. Ahmed, A. T. Azar, Z. Haider, and I. K. Ibraheem, "Hybrid Artificial Fish Swarm Optimization with Deep Learning-Driven Cloud Assisted Cyberattack Detection," International Journal of Intelligent Engineering & Systems, vol. 17, no. 4, 2024.
[14] M. Bakro et al., "Building a cloud-IDS by hybrid bio-inspired feature selection algorithms along with random forest model," IEEE Access, vol. 12, pp. 8846-8874, 2024.
[15] S. Sumathi and R. Rajesh, "A dynamic BPN-MLP neural network DDoS detection model using hybrid swarm intelligent framework," Indian J. Sci. Technol, vol. 16, no. 43, pp. 3890-3904, 2023.
[16] A. Jain, R. Nayak, and U. Ghugar, "Analysis and Mitigation of DDOS Vulnerabilities in Cloud Computing," in 2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), 2023: IEEE, pp. 1-7.
[17] R. Benni, N. Spandana, V. S. Ganagi, and U. Hiremath, "Swarm-Based Approaches for Network Security to Detect and Optimize DOS and DDOS Attack," in Congress on Intelligent Systems, 2024: Springer, pp. 653-677.
[18] A. Shrivastava and P. Gautam, "In Cloud Computing Detection of DDoS Attack Using AI-Based Ensembled Techniques," in International Conference on Communications and Cyber Physical Engineering 2018, 2023: Springer, pp. 1001-1012.
[19] A. H. Reddy, V. S. Reddy, and D. S. Reddy, "Optimized and Intelligent IDS for detecting the vulnerability of DDoS attacks in cloud environment," Journal of Computational Analysis & Applications, vol. 33, no. 6, 2024.
[20] D. Srilatha and N. Thillaiarasu, "DDoSNet: A deep learning model for detecting network attacks in cloud computing," in 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), 2022: IEEE, pp. 576-581.
[21] M. Arunadevi and V. Sathya, "DDoS Attack Detection using Optimized Back Propagation Neural Network with Artificial Plant Optimization in Cloud Computing," in 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), 2022: IEEE, pp. 815-820.
[22] A. Ali, "Detection and Prevention of Distributed Denial of Service (DDoS) Attacks using Metaheuristic and Machine Learning Techniques," Int. J. Sci. Res., vol. 13, p. 10, 2024.
[23] S. Naiem et al., "DDOS attacks defense approaches and mechanism in cloud enviroment," Journal of Theoretical and Applied Information Technology, vol. 100, no. 13, pp. 4632-4642, 2022.
[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.
[25] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," International journal of swarm intelligence, vol. 1, no. 1, pp. 36-50, 2013.
[26] Z. Hu, Y. Bao, and T. Xiong, "Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization," in 2014 IEEE Congress on Evolutionary Computation (CEC), 2014: IEEE, pp. 2259-2265.
[27] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in 2015 military communications and information systems conference (MilCIS), 2015: IEEE, pp. 1-6.

[28] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," ICISSp, vol. 1, pp. 108-116, 2018.
[29] K. Velusamy and R. Amalraj, "Cascade correlation neural network with deterministic weight modification for predicting stock market price," in IOP Conference Series: Materials Science and Engineering, 2021, vol. 1110, no. 1: IOP Publishing, p. 012005.
[30] K. Velusamy and R. Amalraj, "Performance of the cascade correlation neural network for predicting the stock price," in 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2017: IEEE, pp. 1-6.