CHAPTER 6

PANTHERA LEO OPTIMIZED MULTILAYER FEED FORWARD LEARNING FOR MULTIPLE DDOS ATTACK DETECTION

6.1 Introduction

This chapter taking into consideration the limitations discussed on the previous contribution proposes the use of a Multi Feedforward Layer Network (MLFFN) in which feature selection and the classification will be reinforced with Panthera Leo Optimization (PLO) methods. With these proposed frameworks in place, the envision to achieve improvements in accuracy together with the decrease in computational time among the various advantages in the case of multiple DDoS attacks detection.

Multiple DDoS attacks are characterized by a coordinated use of several attack techniques to overload a target system, network, or service, with the potential to cause service outages, financial losses, and harm to an organization's reputation (Zhou, L et al., 2023). This has to be done in a more sophisticated way, which makes handling or addressing it very challenging. The stability and security of networks worldwide are therefore at risk due to the frequent occurrence of DDoS assaults. Recognizing and thwarting various DDoS assaults is a crucial goal to guarantee the functioning and security of the online services and assets. If such attacks are detected and dealt with early, the businesses will mainly experience low availability loss, data loss, and stakeholder trust erosion. Furthermore, detection of several attacks offers opportunity to undertake proper precautions to enforcing the protection layers against the further attacks thus maintain the continuity of operations and improve the organizational cybersecurity (Koay, A et al., 2018) (Zhao, Z, et al., 2024).

Therefore, it is imperative that an organization of work of specialized IDS is built and implemented from the ground up and utilized PLO feedforward learning machines in an ensemble setting. This strategy focuses on identifying malicious nodes and aims to enhance the capacity to foresee assaults. This research also demonstrates the use of a number of hazardous conditions to assess the appropriateness of the suggested approach. The ensembled PLO- learning algorithm based IDS deployment is presented in this study. It has also been shown that this approach is dependable and maintains consistent detection efficiency irrespective of the quantity of hostile nodes.

Most of the previously described issues with current systems are successfully resolved by the novel and creative structure of the appropriate IDS with optimized extreme learning machines that are proposed. The following is the contribution of the recommended techniques in this phase:

(i). that is why developing Highly Effective Ensembled Feed-Forward Learning-based IDS that can of working with larger attack datasets is possible.

(ii). The CICDDOS2019 which is comprised of 80 characteristics and attack types is used to test the proposed technique.

(iii). Since various performance measures can be computed and analyzed with the two given datasets, the experiment is carried out using them.

6.2 Proposed Framework and approach

A multi-DDoS intrusion detection system is developed utilizing the third contribution of this thesis Panthera Leo Optimized Multilayer Feed Forward Learning. This technique combines superior optimization methods with deep learning for a greater identification of multi-vector DDoS attacks. This approach also employ the multilayer feedforward neural network trained by means of the modified algorithm, the so-called Panthera Leo Optimization (PLO), which helps effectively differentiate between several DDoS attack kinds. Through efficient feature selection and parameter tweaking, PLO which is based on lion hunting tactics is included into the procedure, improving detection accuracy and reducing computing complexity.

The employed methods include the multilayer feedforward neural network during the deep learning process with focus on the ability to identify intricate patterns within the accumulated data (Ghanem, W.A. and Jantan, A., 2020) and the refinement of the feature selection in addition to the enhancement of various parameters using the Panthera Leo Optimization algorithm. This combination improves the hope of optimal performance of the detection system which needs high precision and recall for multi-form DDoS attack vector. The use of the proposed approach will have many advantages:

(i) Enhanced Accuracy: Through adjustment of the parameters of a neural network, the IDS manages to work for identifying different types of DDoS attacks with better accuracy.

(ii) Reduced False Positives: A good feature set helps in reducing the number of false positives that is, it doesn't identify the actual trespassing traffic.

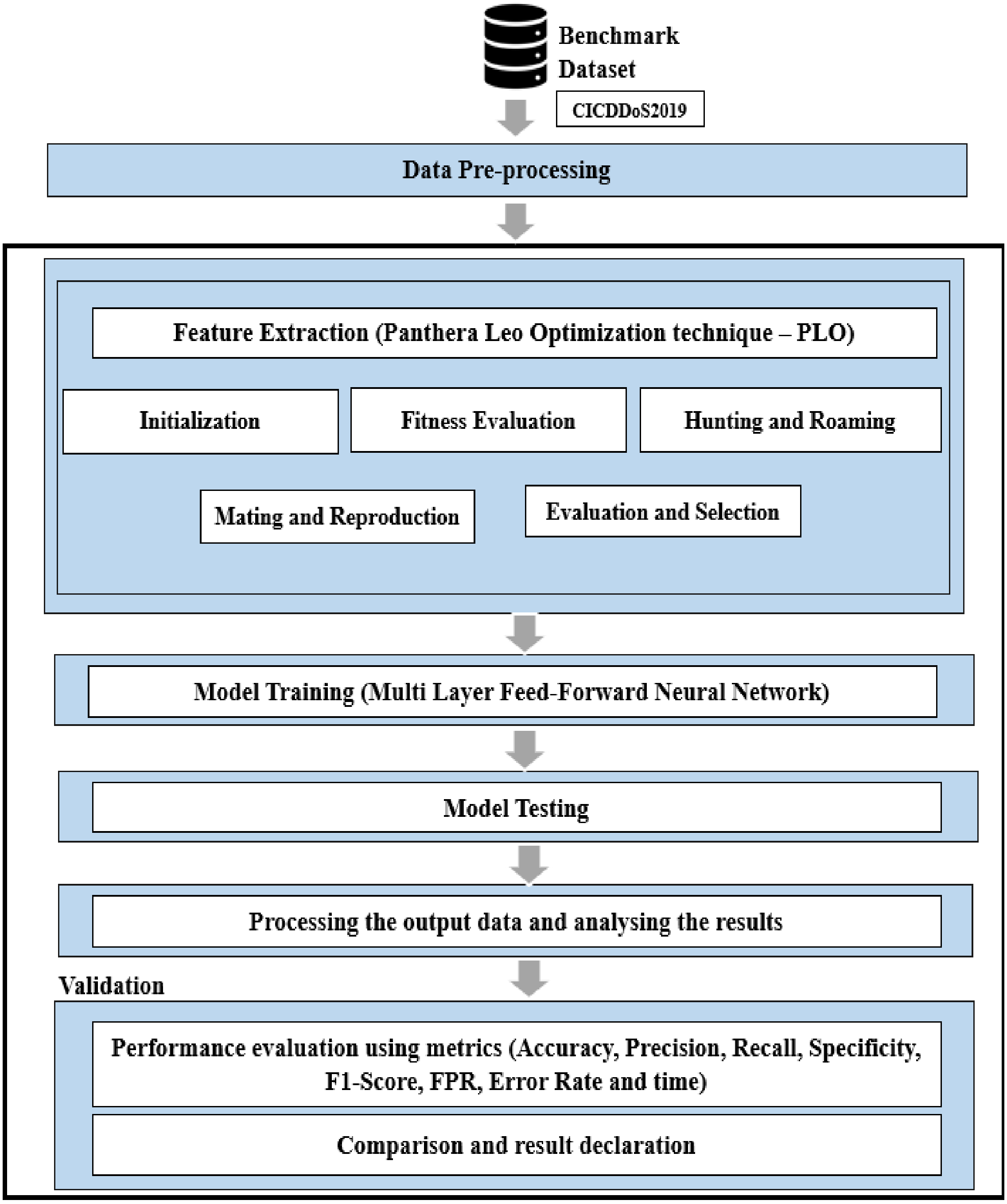

Figure 6.1 Framework for Panthera Leo Optimized Multilayer Feed Forward Learning for Multiple DDoS Attack Detection in Phase III

Essentially, it involves exposing the neural network to labeled input data so that it can distinguish between DDoS assaults and regular traffic. Weights and biases must be adjusted during network training using the PLO algorithm in order to reduce classification error. Figure 6.1 shows how the Panthera Leo Optimized Multilayer Feed Forward Learning approach, a novel feed forward learning technique, is assessed against the modern algorithms to analyze its efficacy (Zulhilmi, A et al., 2021) (Singh, G. and Khare, N., 2022) (A. -C. Enache and V. Sgârciu, 2014). When the suggested approach is contrasted with other approaches, it shows improved accuracy, precision, and recall values, indicating its efficiency in identifying multi-vector DDoS attacks.

The selection of the PLO algorithm in combination with the MLFFN is driven by the need for a balance between model complexity and efficient detection performance. PLO offers a nature-inspired optimization strategy that excels in both exploration and exploitation of the feature space, allowing it to fine-tune network parameters and select the most relevant features effectively. This results in improved convergence and detection accuracy while keeping computational requirements within acceptable bounds. The MLFFN component complements this by enabling the model to learn complex patterns present in diverse DDoS traffic data. Compared to traditional optimization methods like genetic algorithms or PSO, PLO demonstrates better adaptability and speed, making it a suitable choice for multi-vector DDoS detection tasks where both detection precision and execution efficiency are crucial.

6.2.1 Dataset

The CICDDoS2019 dataset includes different DDoS attack types and different attack patterns with many variations. It contains direct network traffic coupled with voluminous flow-based attributes which makes it an information-intensive dataset. Additionally, it uses the CICDDoS2019 dataset to offer a more thorough perspective of the many types of DDoS assaults, including HTTP-based, TCP-SYN, and UDP-based attacks. It is appropriate to utilize for the verification of the suggested Panthera Leo Optimized Multilayer Feed Forward Learning technique and ensures that it is resistant to various DDoS assaults due to its real-life application and subject comprehensiveness. The suggested detection technique may reach a high level of relevance and dependability by utilizing this dataset (Khalid, S., Khalil, T. and Nasreen, S., 2014).

6.2.2 Feature Extraction (Panthera Leo Optimization technique – PLO)

To obtain the best collection of features from a given dataset, the Panthera Leo Optimization (PLO) method integrates lion hunting strategies into feature extraction. The idea of PLO is to enhance the model classification function by identifying the best characteristics that aid in differentiating one class or category from another. The lions, a population of potential feature subsets, are created at random throughout this procedure. A feature subset is represented by each lion, which indicates if a feature is present or not based on a binary value. A classification method is also used to assess the Lion's fitness. After then, lions are divided into two groups: nomadic and pride. Nomadic lions travel different areas of the feature space, while pride lions focus on eating the local areas. Lions cooperate and contend to select the fine feature subset that optimizes classification accuracy while lowering redundancy and overfitting via territorial defense, pride-centric mating, migration, and nomadic mating. Until a termination criterion is met like achieving a maximum number of iterations or a desired level of performance, this iterative process keeps going (Boothalingam, R 2018) (Yazdani, M. and Jolai, F., 2016).

Table 6.1 Pseudocode representation of the Panthera Leo Optimization (PLO) algorithm

```
function PantheraLeoOptimization(num_features, num_iterations, num_lions):
    Initialize population of lions randomly
    Evaluate fitness of each lion using objective function
    Select the top lions based on fitness to form the pride

    for iteration = 1 to num_iterations:
        for each lion in pride:
            Explore surroundings and update position
            Evaluate fitness of updated position
            if fitness improved:
                Update lion's position

        Select the top lions based on fitness to form the new pride
        if new pride is better:
            Replace old pride with new pride
        else:
            Randomly replace some lions in old pride

        Update exploration rate based on iteration
```

The pseudocode representation of the Panthera Leo Optimization (PLO) algorithm for feature set extraction is seen in the table 6.1. Here, a two-fold data split of 50% for training and 50% for validation is used, and PLO is applied in incremental phases to pick features that provide the best classification accuracy for the chosen classifier. The locations of the lions that correspond to the feature subsets must be adjusted throughout the PLO algorithm's iterations to measure the fitness of the candidate feature using a classifier indicator (Ravindranath, V et al., 2020). Repeating this procedure until a termination condition is achieved; the most performing feature subset is then used for other purposes, including DDoS detection systems. The PLO process starts with below steps.

Step 1: Creation of an initial population of lions which involves generation of each lion of the population carrying with it a set of features. The lions are symboled as xi each is a vector of 0s and 1s indicating the chosen features and their fitness is estimated using a predetermined function.

- Lion Representation: Let *xi=[xi1,xi2,...,xin]* represent the i_{th} lion, where xij∈{0,1} indicates whether the j-th feature is selected.
- Fitness Function: The fitness function *f(xi)* measures the feature subset xi's performance using a ML model.

Step 2: Pride and Nomad Lions involves dividing the lion population into two groups: prides and nomads. Prides explore the search space broadly, while nomads focus on refining the best-known solutions.

- Divide the lion population into prides P={x1,x2,...,xk} and nomads N={xk+1,xk+2,...,xm}. Prides explore the search space broadly, while nomads exploit the best-known solutions.

Step 3: Hunting and Roaming, lions in prides engage in a hunting update, adjusting their position based on the best-known feature subset. This step includes a roaming phase for nomads, where new feature combinations are explored through random changes.

- Hunter Update: $x_i^{t+1} = x_i^t + \alpha.(x_{best}^t - x_i^t)$

α is a random number [0,1], x_{best}^t is the best-known feature subset at time t

- Roaming Update: $x_i^{t+1} = x_i^t + \beta.randn()$

- β is a step size parameter
- randn() generates random values

Step 4: Mating and Reproduction has lions producing offspring by combining features from two parent lions. The offspring generation is performed using a crossover mechanism.

- Offspring Generation: $x_{offspring} = crossover(x_{parent1}, x_{parent2})$
- Employ crossover strategies like uniform or single-point crossover

Step 5: Evaluation and Selection evaluates the fitness of each lion and selects the best feature subsets for the next generation. The fitness of every subset is measured and lions with the greatest fitness scores are retained.

- Fitness Evaluation: f(xi) for each lion xi
- Selection: Retain lions with the highest fitness values

The pseudocode for the proposed approach in Phase III outlines the detailed stages of the PLO algorithm as shown in Table 6.2. The first step is to initialize the population and neural network parameters, followed by the main optimization loop which includes hunting, training, and evaluating neural networks with feature subsets from prides and nomads. Mating and reproduction steps follow, generating offspring and evaluating their fitness. The best feature subsets are then selected and combined for further iterations until the optimal solution is achieved.

Table 6.2 Pseudocode for the proposed approach in Phase III

Step 1. Initialize parameters: ***- Population size (N)*** ***- Maximum number of iterations (MaxIter)*** ***- Dimensionality of the problem (D)*** ***- Convergence criteria (e.g., minimum improvement in fitness)*** ***- Neural network architecture*** ***Step 2. Initialize population of lions (solutions):*** ***For each lion i = 1 to N:***

Initialize random values for weights and biases

Step 3. Define fitness evaluation function:

Function evaluate_fitness(lion):

Assign weights and biases according to lion's parameters

Train the neural network on training data

Compute fitness based on validation set accuracy or mean squared error

Step 4. Main loop:

Initialize iteration counter (Iter) = 0

While Iter < MaxIter:

For each lion i = 1 to N:

Evaluate fitness: fitness[i] = evaluate_fitness(lion[i])

Sort lions based on fitness values in descending order

For each lion i = 1 to N:

Perform PLO movements to update lion's parameters (weights and biases):

- Searching

- Encircling prey

- Ambushing

Update position (parameters) of lion[i] based on PLO movement

Increment iteration counter: Iter = Iter + 1

Check convergence criteria:

If improvement in fitness is below threshold or other criteria are met, break loop

Step 5. Select the best lion (solution) based on fitness

Step 6. Deploy and validate the optimized neural network:

Apply the parameters of the best lion to configure the neural network

Evaluate functionality on a independent test set

Step 7. Output the optimized neural network model and its performance metrics

Benefits of PLO Techniques

(i) More exploration and exploitation than other meta-heuristic algorithms like PSO, genetic algorithms, and even the Grey Wolf optimizer (Yazdani, M. and Jolai, F., 2016).

(ii) Less time complexity.

6.2.3 Proposed Model Training and Testing

To lessen the difference between the expected output and the real target values, the Multi Feed-Forward Neural Network's parameters are iteratively adjusted during model training (Rosay A et al., 2020) (Kshirsagar, P.R., Yadav, R.K. and Patil, N.N., 2022). This procedure begins by initializing the network's weights and biases in a standard way using any random number. The network is then given the training data, which are input features created by the data, and the target labels. After that, the network computes an error by comparing its output to the target labels; this is often accomplished using a regression loss function. The method known as back propagation is used to minimize the aforementioned problem by taking the error from the final output and extending it to every network layer. This process keeps on until the whole input set has been run through the networks several times, or epochs, until the networks' weights and biases have been progressively adjusted to provide the optimal outcomes. In order for a network to effectively tackle invisible issues, model training involves determining the best set of criteria’s to capture the underlying structure and learning in the data. Table 6.3 describes the parameter configuration for the suggested network.

Table 6.3 Settings for the Proposed PLO-MLFFN Parameters

Parameter	Description	Sample Value/Options
Input Layer Size	Number of input features	9
Hidden Layers	Number of hidden layers and neurons per layer	2 layers with 64 neurons each
Output Layer Size	Number of output classes or units	6 (multi-class classification)
Activation Function	Function applied to the neurons in each layer	ReLU
Loss Function	Function to compute the error/loss during training	Cross-Entropy
Optimizer	The training optimization algorithm	Panthera Leo Optimization Algorithm
Learning Rate	Step size for weight updates when training	0.001
Batch Size	The quantity of training samples in every batch	32
Epochs	The number of times the whole training dataset is processed	100

6.3 Experimental Setup

The whole experimental configuration includes a 256 GB Nvidia Titan board, 16 GB of RAM, and an i9 CPU.

6.4 Experimental Results and Discussion

The efficacy of the suggested work is shown and examined in the part that follows when the PLO Algorithm and MLFFN are used together. The CICDDoS2019 dataset is used for the experiments. In this instance, the objective is to assess how well the combined approach performs in improving feature selection and identification accuracy. The purpose of the findings reported here is to get a better understanding of the effectiveness of the PLO-MLFFN architecture and its likely uses in the cybersecurity domain. Using the performance matrix the suggested model's quantitative performance is evaluated. Figure 6 displays the Table of Confusion for the suggested model. The performance of a suggested method to differentiate between several network attacks types with varying percentages on the diagonal showing the extent of correct predictions. In addition, the aerial design is evaluated against other existing models by forecasting numerous display quantities.

Confusion matrix for the Proposed algorithm

Predicted Values

Actual Values

Attacks	UDP Flood	TCP SYN Flood	HTTP GET Flood	DNS Amplification Attack	NTP Amplification Attack	UDP Flood + HTTP GET Flood	TCP SYN Flood + HTTP POST Flood
UDP Flood	96.80%	0.50%	0.50%	0.50%	0.50%	0.60%	0.60%
TCP SYN Flood	0.50%	96.70%	0.50%	0.50%	0.60%	0.60%	0.60%
HTTP GET Flood	0.50%	0.60%	96.75%	0.50%	0.60%	0.60%	0.60%
DNS Amplification Attack	0.50%	0.50%	0.50%	96.85%	0.60%	0.50%	0.50%
NTP Amplification Attack	0.50%	0.50%	0.60%	0.50%	96.78%	0.60%	0.50%
UDP Flood + HTTP GET Flood	0.60%	0.60%	0.50%	0.50%	0.60%	96.80%	0.50%
TCP SYN Flood + HTTP POST Flood	0.50%	0.60%	0.50%	0.50%	0.50%	0.60%	96.80%

Figure 6.2. Phase III Confusion Matrix for the Proposed Model

The confusion matrix compares the true attack labels to the predicted labels. Every row indicates the actual attack type and every column the predicted attack type. Values on the diagonals (green boxes) represent correct classifications while the boxes on the off-diagonals represent misclassifications. In each cell, the percentage values illustrate the ratio of samples attributed to each prediction category, allowing to get a information of the model's accuracy and mistakes.

The findings demonstrate that for almost every kind of assault, the model attains an acceptable classification accuracy. In particular, UDP Flood, TCP SYN Flood, and HTTP GET Flood assaults have respective accuracies of 96.80%, 96.70%, and 96.75%. Likewise, the accuracy of the DNS amplification and NTP amplification attacks was 96.85% and 96.78%, respectively. Regarding combination assaults like UDP Flood + HTTP GET Flood and TCP SYN Flood + HTTP POST Flood the accuracy remained high 96.80% for both cases. High values along the diagonal imply that the model effectively distinguishes between different attack types, even including complex, multi-attack combinations.

Despite the overall strong performance, the table also reveals minor misclassifications. Off-diagonal numbers are between 0.50% and 0.60%, so there is a little confusion between the tom categories. For example, the UDP Flood was incorrectly classified as TCP SYN Flood and HTTP GET Flood only 0.50% of the time.

To summarize, The model's ability to accurately forecast network assaults in spite of hybrid attack scenarios is shown by the confusion matrix. However, the model's low misclassification rates show that it is reliable. The need of creating classification systems that can precisely identify both single and coordinated threats is highlighted by this investigation. In order to prove the suggested IDS's superiority, this study compared it to other cutting-edge existing IDS at different training dataset proportions.

The classification approaches validated in this chapter are,

(i) SVM

(ii) CART

(iii) ELM

(iv) BAT-Optimized ELM – (BAT-Optimized ELM)

(v) Panthera Leo Optimization - Multi-Layer Feedforward Neural Network - (Proposed PLO-MLFFN)

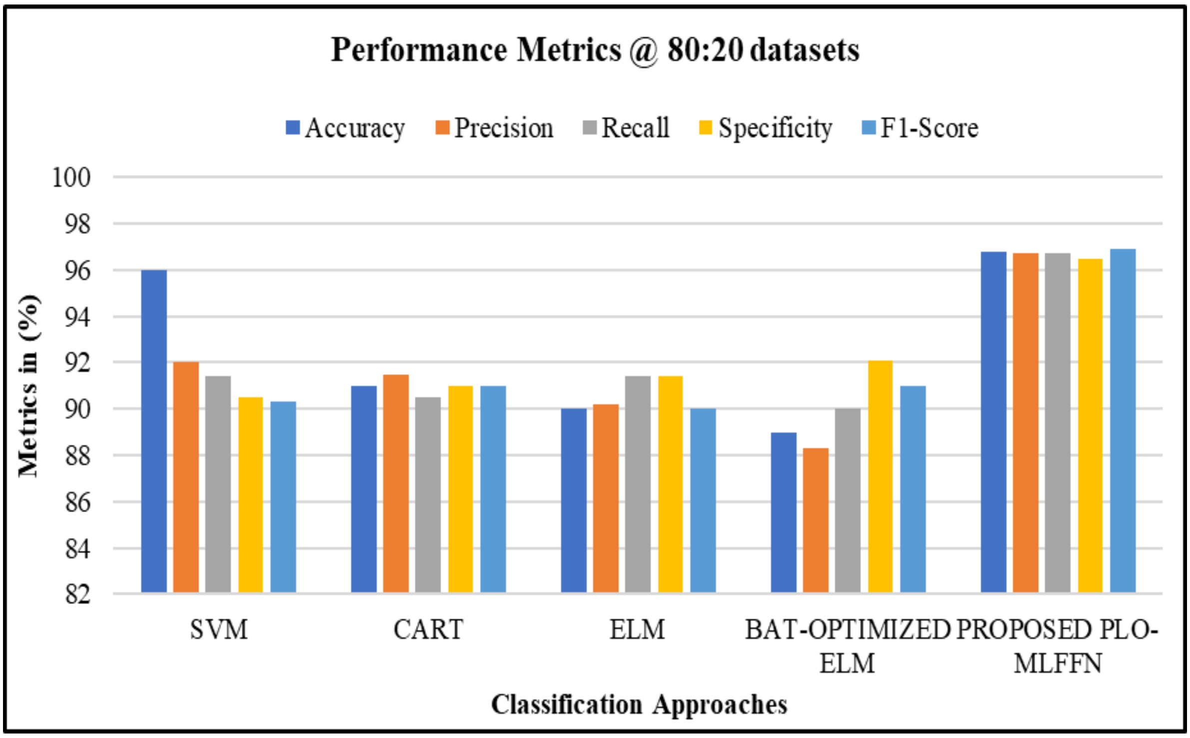

Figure 6.3 Comparative Evaluation of the Various IDS at the 80:20 dataset ratio

Table 6.4 Metric values of different IDS at the ratio of 80:20

Performance Metrics @ 80:20 datasets					
Metrics/Approaches	**SVM**	**CART**	**ELM**	**BAT-OPTIMIZED ELM**	**PROPOSED PLO-MLFFN**
Accuracy (%)	96	91	90	89	**96.8**
Precision (%)	92	91.5	90.2	88.3	**96.75**
Recall (%)	91.4	90.5	91.4	90	**96.73**
Specificity (%)	90.5	91	91.4	92.1	**96.5**
F1-Score (%)	90.3	91	90	91	**96.89**

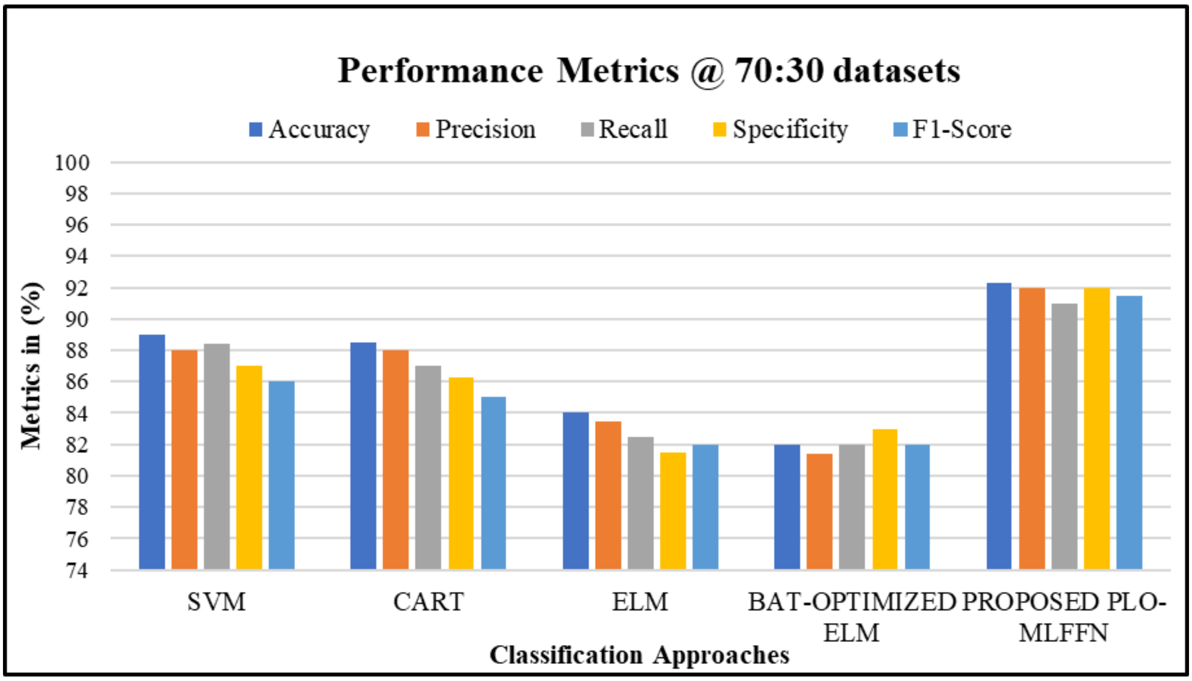

Figure 6.4 Comparative Evaluation of the Various IDS at the 70:30 dataset ratio

Table 6.5 Metric values of different IDS at the ratio of 70:30

Performance Metrics @ 70:30 datasets					
Metrics/Approaches	**SVM**	**CART**	**ELM**	**BAT-OPTIMIZED ELM**	**PROPOSED PLO-MLFFN**
Accuracy (%)	89	88.5	84	82	**92.3**
Precision (%)	88	88	83.5	81.4	**92**
Recall (%)	88.4	87	82.5	82	**91**
Specificity (%)	87	86.3	81.5	83	**92**
F1-Score (%)	86	85	82	82	**91.5**

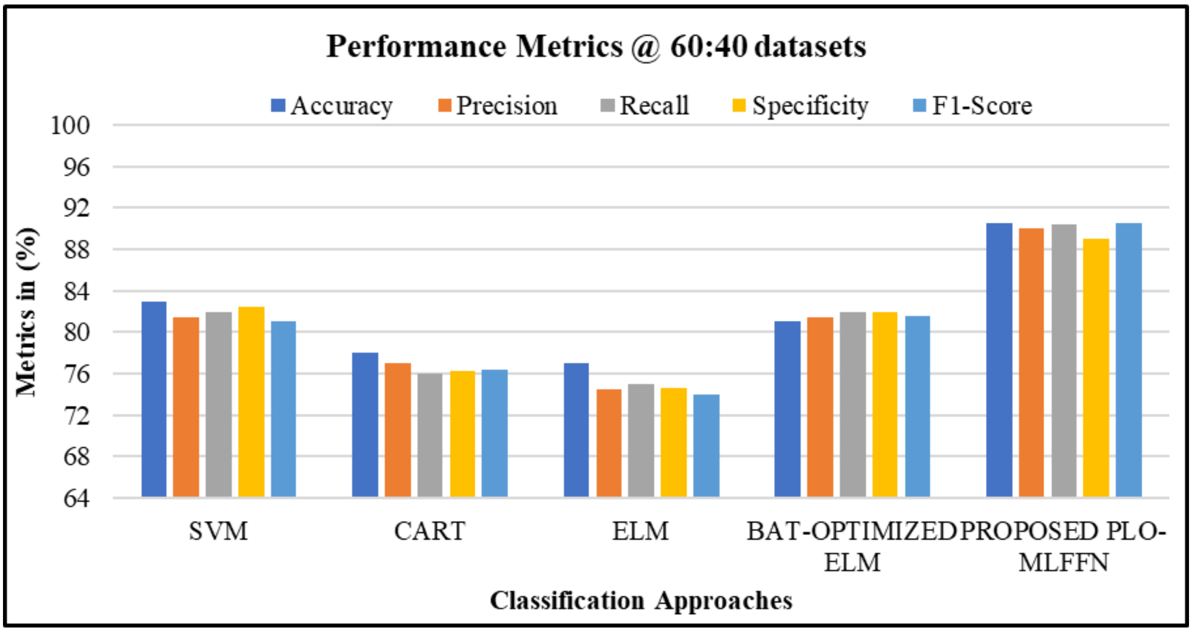

Figure 6.5 Comparative Evaluation of the Various IDS at the 60:40 dataset ratio

Table 6.6 Metric values of different IDS at the ratio of 60:40

Performance Metrics @ 60:40 datasets					
Metrics/Approaches	**SVM**	**CART**	**ELM**	**BAT-OPTIMIZED ELM**	**PROPOSED PLO-MLFFN**
Accuracy (%)	83	80.6	77	81	**90.5**
Precision (%)	81.4	79.6	74.5	81.4	**90**
Recall (%)	82	78	75	82	**90.4**
Specificity (%)	82.4	78.9	74.6	82	**89**
F1-Score (%)	81	79	74	81.5	**90.5**

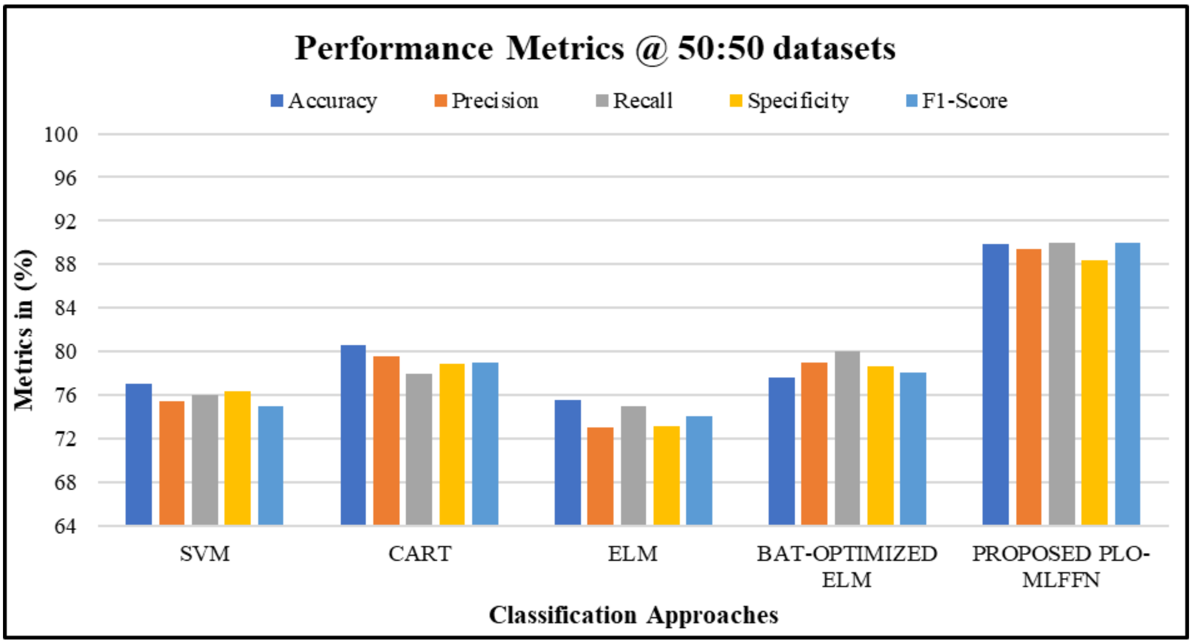

Figure 6.6 Comparative Evaluation of the Various IDS at a 50:50 dataset ratio

Table 6.7 Metric values of different IDS at the ratio of 50:50

Performance Metrics @ 50:50 datasets					
Metrics/Approaches	**SVM**	**CART**	**ELM**	**BAT-OPTIMIZED ELM**	**PROPOSED PLO-MLFFN**
Accuracy (%)	77	78	75.58	77.6	**89.9**
Precision (%)	75.4	77	73.08	79	**89.4**
Recall (%)	76	76	75	80	**90**
Specificity (%)	76.4	76.3	73.18	78.6	**88.4**
F1-Score (%)	75	76.4	74	78.1	**90**

Figures 6.3 to 6.6 and table 6.4 to 6.7 present a comparative analysis of various algorithms alongside a variable dataset. These figures clearly demonstrate that the proposed IDS maintains consistent performance even as the count of malicious nodes rises, whereas other ML-based IDS show worsened performance. Thus, it is clear that the suggested IDS functions far better than its peer learning-based IDSs when dealing with a higher number of malicious nodes. Additionally, the proposed algorithm's ability to maintain constant

detection time during data split testing (80:20, 70:30, 60:40, and 50:50) arrangements are far more valuable as they identify the specific value-addition that was lacking. It shows how fast the algorithm is when calculating the entropy and how unimpeachable it remains irrespective of the input data distribution. Such stability is important for ensuring the timely and appropriate detection and response to threats, leading to improvement of IDS security and performance in dynamic and busy networks. The performance of some of the algorithms at different proportions of the dataset is summarized in table 6.7.

Table 6.8 Comparative Evaluation of the Various IDS with Regard to Effective Attack Processing Time

S.No	Algorithms	Time(seconds) – per epoch			
		80:20	70:30	60:40	50:50
1	SVM	9.4	12.3	13.5	14.3
2	CART	10.4	13.0	14.2	15.3
3	ELM	11	13.2	14.3	15.3
4	BAT – Optimized ELM	9.1	9.1	9.0	8.9
5	**PROPOSED PLO-MLFFN**	**7.6**	**7.6**	**7.5**	**7.5**

The table 6.8 presents a comparison of the time taken per epoch (in seconds) by different machine learning algorithms under varying data split ratios for testing and training: 50:50, 60:40, 70:30, and 80:20. The algorithms that were assessed include SVM, CART, ELM, BAT-Optimized ELM, and the Proposed PLO-MLFFN.

Among the other methods, SVM shows increasing time requirements as the data split becomes more balanced, starting at 9.4 seconds for an 80:20 split and rising to 14.3 seconds for a 50:50 split. Similarly, CART and ELM exhibit slightly higher times compared to SVM, with CART ranging from 10.4 to 15.3 seconds and ELM from 11 to 15.3 seconds. In contrast, the BAT-Optimized ELM demonstrates more consistent and lower times across splits, ranging between 9.1 and 8.9 seconds, indicating better efficiency than the other approaches.

The Proposed PLO-MLFFN outperforms all other algorithms, achieving the lowest and most consistent times, ranging from 7.6 to 7.5 seconds. This result highlights the efficacy and scalability of the proposed methodology, especially as the data split ratio becomes more balanced, making it a promising solution for time-sensitive applications.

The comparison of many performance indicators for several DDoS attack types using various models is displayed in Figure 6.7 (a, b, c, d, and e). Supervised Vector Machines, Classification and Regression Trees, Extreme Learning Machines features, BAT Optimized ELMs, and the Proposed Model are the five machine learning models whose performances are evaluated and compared under the various attack types using the set of graphs displayed in Figure 6.7. In addition to differentiating between UDP Flood, TCP SYN Flood, HTTP GET Flood, DNS Amplification Attack, NTP Amplification Attack, and more complicated assaults like UDP Flood + HTTP GET Flood and TCP SYN Flood + HTTP POST, each graph displays the findings for the attacks that were analysed. The findings once again demonstrate that the proposed model outperforms its peers when compared to all metrics and attack types. While the BAT-Optimized ELM also shows good performance, it does not match the Proposed Model’s consistency. In contrast, other methods like SVM, CART, and ELM exhibit lower performance, making them less effective for advanced network attack detection tasks. These results confirm the Proposed Model's strength as a reliable solution for intrusion detection systems.

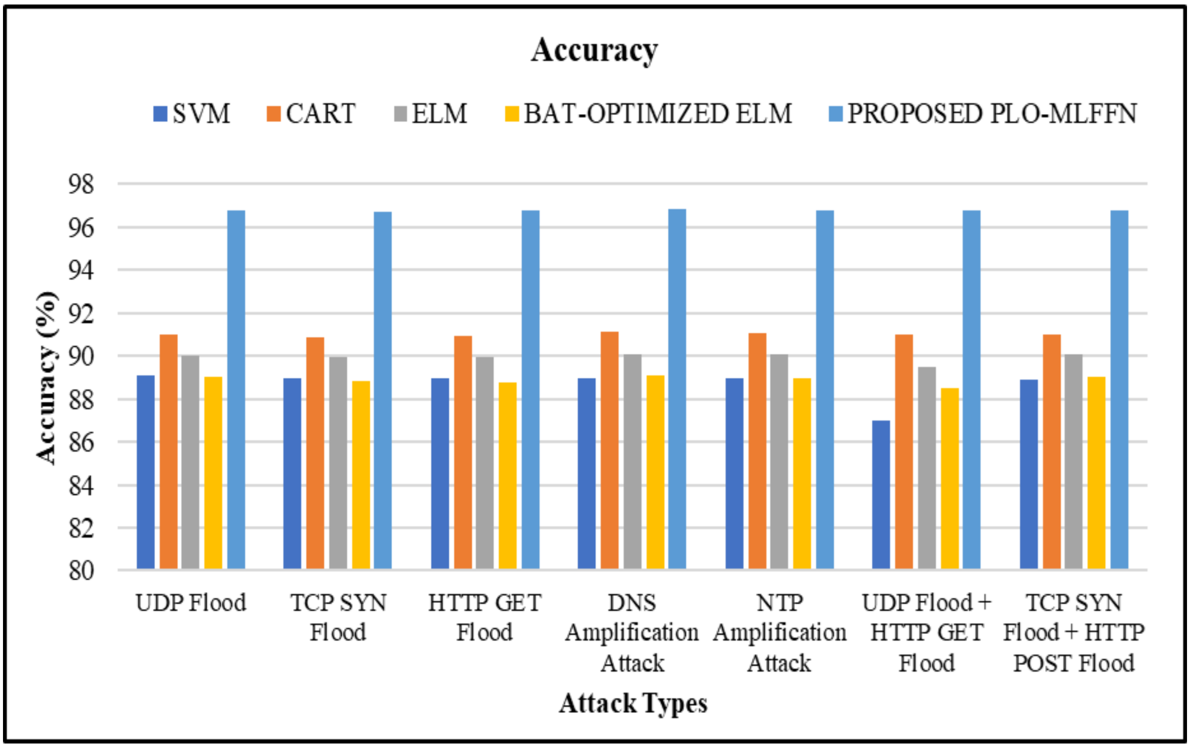

Figure 6.7 (a)

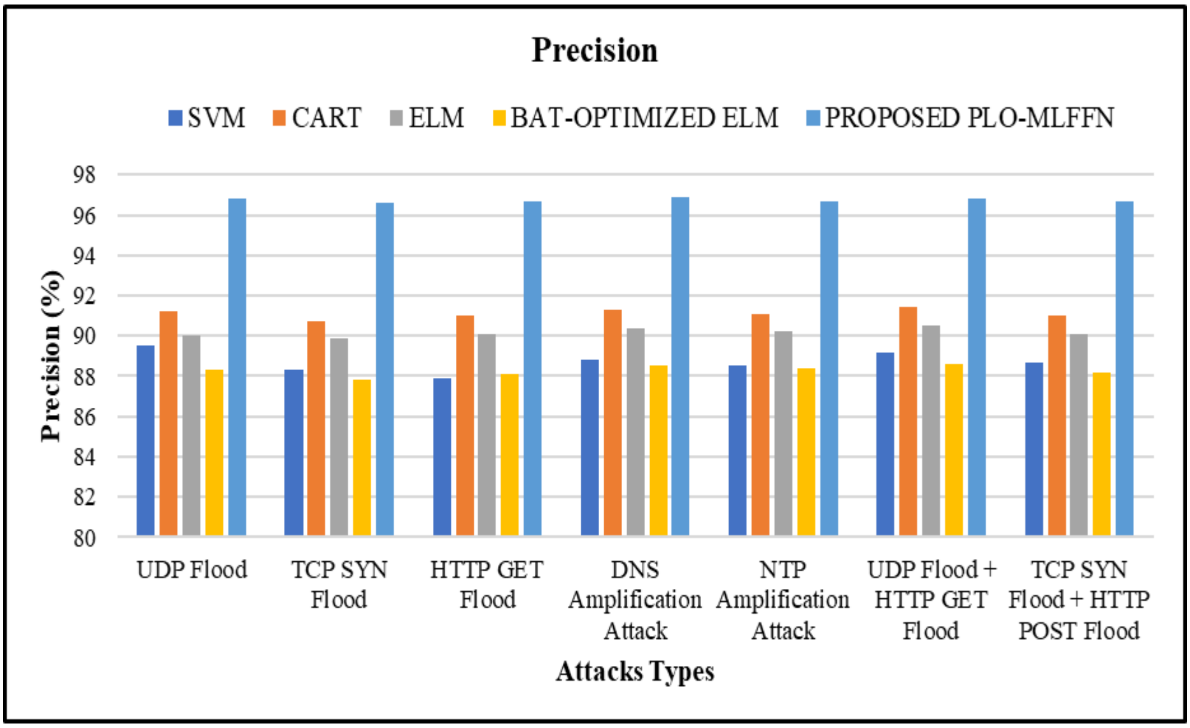

Figure 6.7 (b)

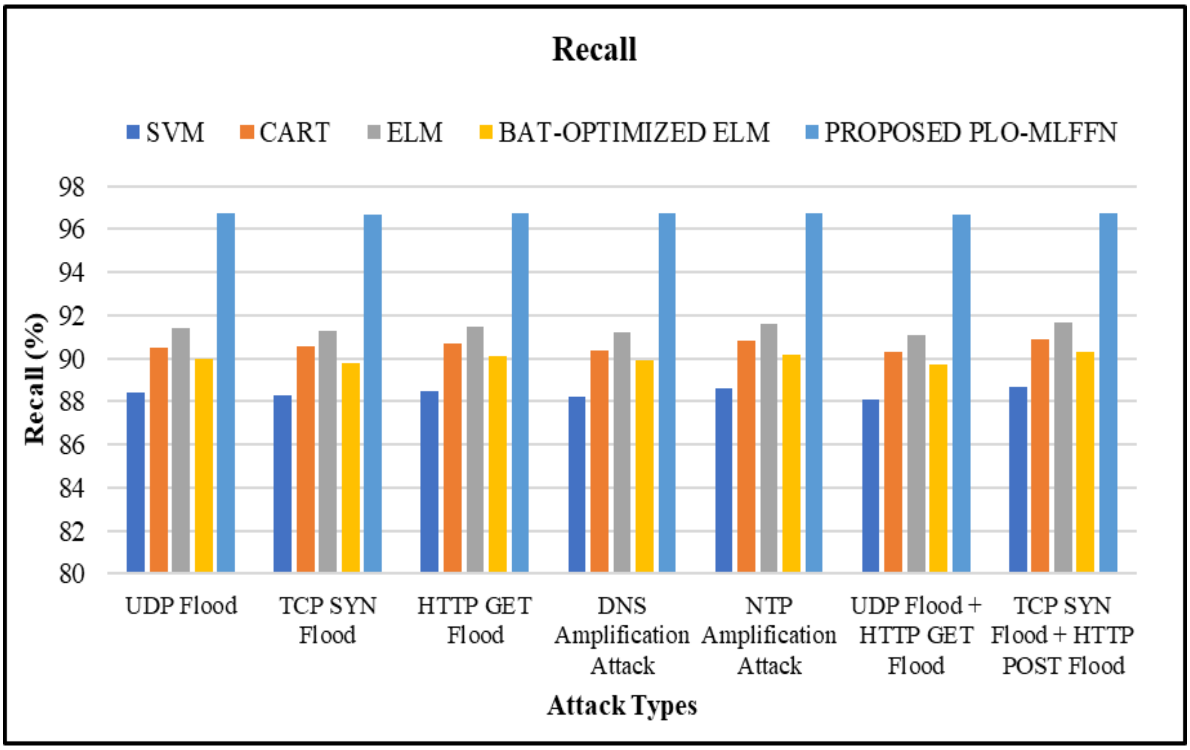

Figure 6.7 (c)

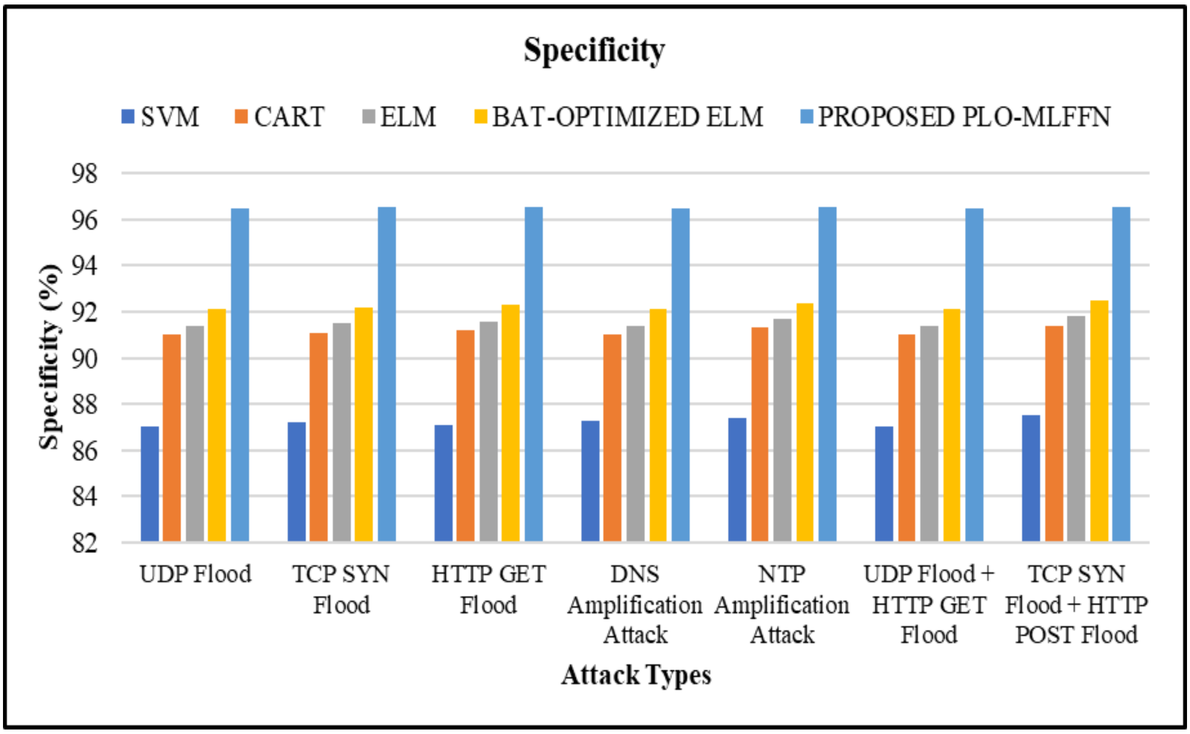

Figure 6.7 (d)

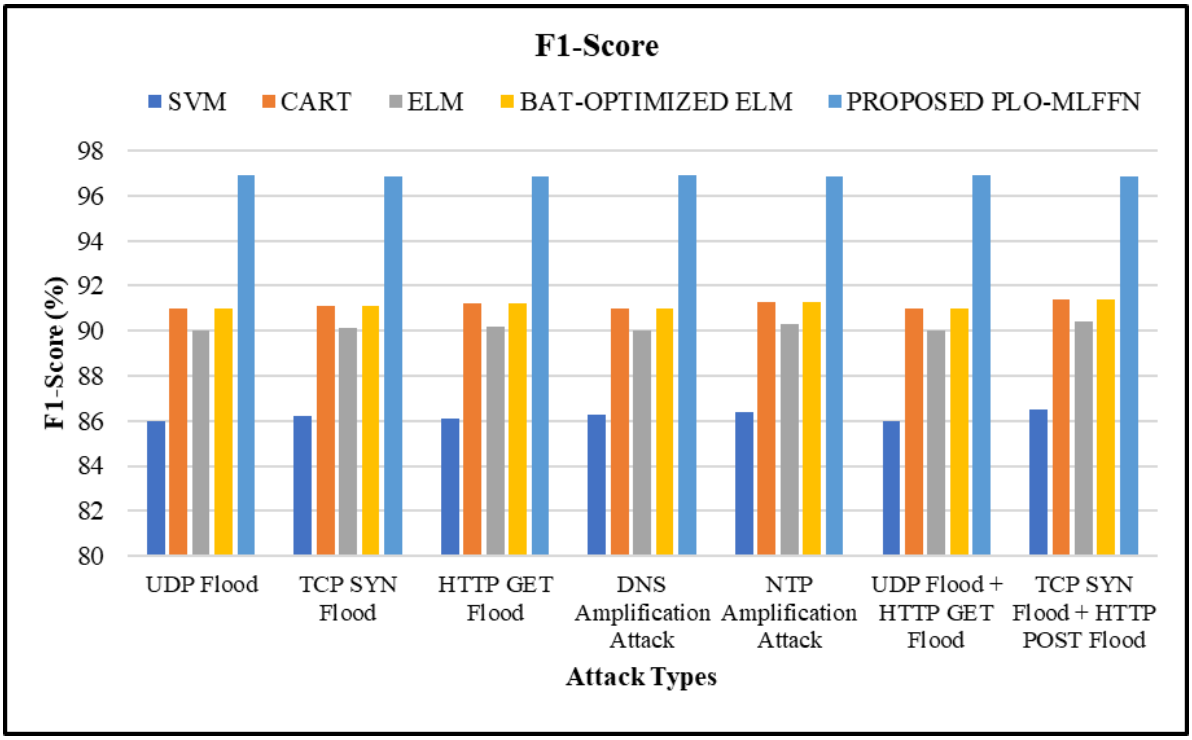

Figure 6.7 (e)

Figure 6.7. Comparative analysis of the different attack types using the proposed method in phase III

6.5 Chapter Summary

In Phase III of the research, the proposed framework integrates Panthera Leo Optimized Multilayer Feed Forward Learning with improved performance, reduced false alarm and scalability in detecting multiple DDoS attacks. Proposed framework achieves 96.8% accuracy, 96.75% precision, 96.73% recall, 96.5% specificity, and 96.89% F1-score for various dataset training and testing ratios. Comparative evaluation of different techniques and datasets continuously shown the suggested IDS's noteworthy performance, despite the fact that the quantity of harmful nodes increased. The PLO-MLFFN algorithm consistently exhibits the lowest detection times across all partition ratios of training and test data (80:20, 70:30, 60:40, and 50:50) compared to other algorithms such as SVM, CART, ELM, and BAT-ELM. This indicates that PLO-MLFFN is the most efficient algorithm with 7.6 secs. The proposed model in Phase III is effective for detecting and dealing with multiple DDoS attacks. This method is complexity-aware, as it adapts to the increasing complexity of the data and evolving attack patterns, ensuring both accuracy and efficiency. Yet, the model requires further development to scale up to large and work with a few different datasets in low computational time and high accuracy.