CHAPTER 7

ATTENTION ENABLED GATED RECURRENT NETWORK (AEGRN) AND DEEP FEED FORWARD NETWORKS FOR DETECTING DDOS ATTACKS IN MULTIPLE DATASETS

7.1 Introduction

This chapter lays the groundwork to examine systematically the efficiency of proposed Attention-Enabled Gated Recurrent Networks (AEGRN) as well as Deep Feedforward Networks to detect DDoS attacks on several datasets. This research work aims to take the detection accuracy and defense against DDoS threats to a new level by using RNNs (Jha, S et al, 2020) enhanced with attention mechanisms and the feedforward network trained in deep learning.

This is because they believe that enhancing the step in feature extraction is vital in increasing the performance and reliability of IDS in distinguishing DDoS attacks (Xu, C et al., 2018). Feature extraction is used to increase the precision of the models by minimizing feature space dimensionality and filtering out noisy attributes most useful in uncovering more intricate patterns and subtle features pointing out to attacks. This results in fewer false positive and negatives making detection a lot more accurate and efficient. Furthermore, detailed feature extraction is useful for increasing model applicability in unknown data, while achieving high-performance across different powering situations, and making real-time detection possible by minimizing computational load. The integration of attention mechanisms into gated recurrent units allows for the dynamic adaptation of attention within the AEGRN model in order to detect subtle indicators of network traffic intrusion (Niu, Z., Zhong, G. and Yu, H., 2021). This attention-driven approach not only enhances the interpretability of the model but also facilitates more precise and targeted detection of DDoS attacks across diverse network environments. Therefore, strengthening feature extraction, combined with advanced mechanisms like attention in AEGRNs, directly contributes to more efficient, scalable, and effective cybersecurity defenses (Qian X et al., 2022) (Li et al., 2020).

Furthermore, the utilization of deep feedforward networks signifies an important change in intrusion detection, leveraging the capacity of DL architectures to retrieve complicated patterns and trends from complex network data. Through the application of

convolutional and fully connected layers, deep feedforward networks possess the capability to autonomously understand and adjust to the changing strategies employed by DDoS attackers (Maithem, M. and Al-Sultany, G.A., 2021).

There are some problems that need to be fixed since these sophisticated IDS have just recently been released. These are some of the issues that malware detection system research is dealing with (Du, Dajun, et al., 2022).

(i) While they usually go low in reaching a very high rate of detection, the bulk of presently used approaches focus on detecting a single assault with a reduced false alarm rate.

(ii) Understanding the various attack characteristics is vital, but it is even more critical to determine which ones may really aid in the detection of assaults. However, certain current methods often have large false positive rates due to duplicated data and excessive computational cost. Because previous approaches also fail to achieve effective accuracy, a well-structured network attack detection technique is still a potential area of study.

In light of these issues, this chapter suggests combining the Gated Recurrent Neural Network (GNN) with Attention layers to get a high classification ratio while reducing computing complexity and thwarting network assaults. The following are the primary aspects of the proposed framework:

(i) To improve feature selection, which in turn supports a higher detection ratio, self-attention maps are added to gate recurrent neural networks.

(ii) Deep Feedforward Learning Layers are introduced to accomplish quicker training while detecting fewer errors.

7.2 Proposed Methodology

According to Figure 7.1, three submodules comprise the structure of the hybrid proposed network. Several pre-processed datasets are sent into the suggested network in the first module. The proposed SA-GRU-FF architecture, which eliminates redundant and suboptimal temporal characteristics by integrating an attention layer, makes up the second module. Deep feedforward networks that are completely linked are then given these attributes for classification of multiple attacks, leveraging techniques inspired by Extreme Learning Machine (ELM) for efficient learning.

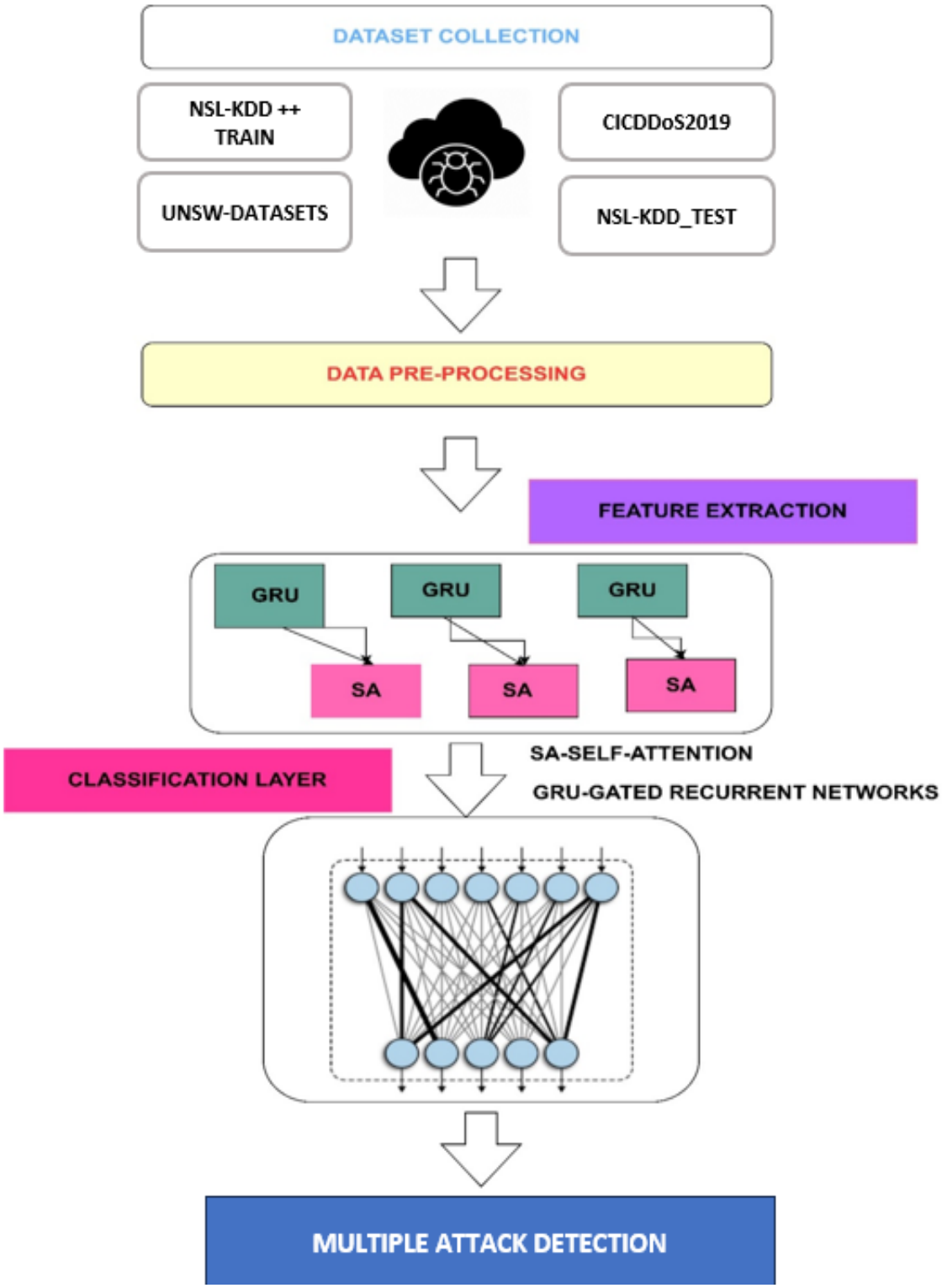


Figure 7.1 Proposed Architecture for the GRU-SA-FF in Phase IV

The selection of the Attention-Enabled Gated Recurrent Network (AEGRN) and Deep Feedforward Networks (DFFN) is based on their complementary strengths in capturing both temporal dependencies and non-linear relationships in network traffic data. GRU-based architectures provide efficient learning for sequential inputs, while the self-attention mechanism enhances focus on critical features, improving the interpretability and detection accuracy across varying DDoS traffic patterns. The DFFN component supports high-speed classification and allows the model to scale effectively across different datasets. This hybrid approach achieves a favorable trade-off by combining deep feature understanding with reduced false positives and computational cost. While deeper models like LSTM and CNNs may offer competitive accuracy, they often suffer from higher training complexity. AEGRN-DFFN provides a scalable, high-performance alternative, particularly well-suited for diverse datasets and real-time IDS applications.

7.3 Materials and Methods

It makes use of three different datasets: CICDDOS2019, UNSW-NB15, and NSLKDD. Recently, the CICDDOS2019 dataset is made available to aid in the creation of anomaly-based IDS. It encompasses a comprehensive range of DDoS assaults and regular network traffic, with data broken down by CICFlowMeter-V3 to record crucial information in CSV files, including timestamps, IP addresses, ports, and attack kinds. It includes approximately 121,980 records of regular traffic and 172,647 records of attacking traffic, covering various DDoS scenarios and updated regularly. The CICDDoS2019 dataset contains 41.4% (121,980 records) classified as normal traffic and 58.6% (172,647 records) categorized as attack traffic. Earlier research (Vinayakumar, R et al., 2019) & (Vinayakumar, R., Soman, K.P. and Poornachandran, P., 2017) examined the effectiveness of various ML classifiers on the CICDDOS2019 dataset. The dataset UNSW-NB15 has one class characteristic in addition to 49 features. There are 82,332 occurrences in the test set and 175,341 in the train set. The train set comprises 119,341 incidents of attack traffic and 56,000 instances of regular traffic. The test set also contains 45,332 incidents of attack traffic and 37,000 cases of regular traffic. The test-only train set is used for cross-fold validation, whereas hold-out validation uses the whole train and test sets.

7.4 Data Reorganizing

A standardization strategy is used to help transform the majority of qualities containing numerical data to a designated numeric domain once the input data has been first

examined. To achieve this, the linear transformation principle is used with Min-Max normalization. The pre-processed datasets files are sent to the feature extraction module.

7.5 Self-Attention Gated Recurrent Networks for Feature Extraction

It discusses self-attention, gated recurrent sections and mixed combinations of self-attention gated recurrent units.

7.5.1 Gated Recurrent Units

GRU is one among the intriguing types of LSTM. According to the idea presented by (Chung, J et al., 2014) (Xu, C et al, 2018) the forget gate and input vector intended to be combined into a single vector. Both long-term memory as well as short-term memory are provided by this network. Comparing the level of complexity with the LSTM network, it is significantly smaller.

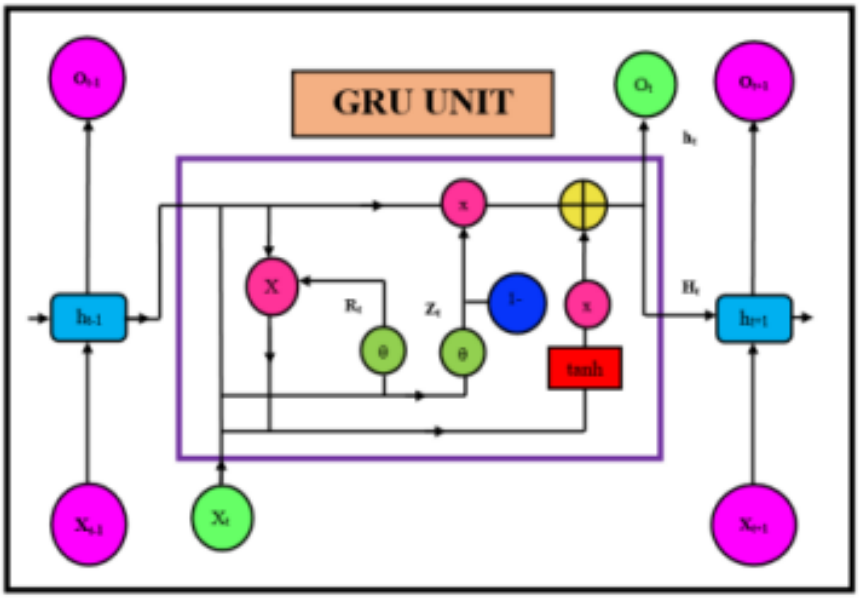


Figure 7.2 Network Architecture (GRU)

The below equations show the properties of GRU (J. Chung et al., 2014).

$$\boldsymbol{h_t = (1 - x_t) \odot h_{t-1} + x_t \odot h_t} \quad \textbf{(7.1)}$$

$$\boldsymbol{\widetilde{h_t} = g(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h} \quad \textbf{(7.2)}$$

$$\boldsymbol{z_t = \sigma(W_h x_t + U_z h_{t-1} + b_z)} \quad \textbf{(7.3)}$$

$$\boldsymbol{r_t = \sigma(W_h x_t + U_r h_{t-1} + b_r)} \quad \textbf{(7.4)}$$

The generic GRU feature equation is as follows:

$$\boldsymbol{R = GRU(\sum_{t=1}^{n}[x_{t,}\, h_{t,} z_{t,} r_t\big(W(t), B(t), \eta(tannh)\big)]} \quad \textbf{(7.5)}$$

where xt ⇨ "input feature at the present state" , yt ⇨ "output state" , ht ⇨ " output of the unit as of this moment", Zt & rt⇨ "update & reset gates", W(t) ⇨ "weights", B(t) ⇨ "bias weights at present instant."

7.5.2 Self-Awareness Maps

The attentive map was introduced (Keller, A., et al 2014) to explain the appropriate terminology in sequence-to-sequence arrangement. The attention layers have been employed in most recent research to mimic redundant aspects that may help with the accurate categorization method. The self-attention procedure, also known as the intra-attention mechanism, produces the three vectors Q, K, and V for every input sequence. Thus, each layer's input sequences are converted to its output sequences. It is a technique that utilizes scaled dot functions to link the query with the set of key pairs. The mathematical computation for the dot product is as follows.

$$\boldsymbol{F(K, Q) = ((K), Q^T))/(V_K)\text{^}0.5} \quad \textbf{(7.6)}$$

7.5.3 Feature Extraction Proposal

GRU networks are constructed as BiGRU networks, which include forward and backward GRU to extract the relevant data from the many sources of the datasets (Bi, J et al., 2023) (Li W et al, 2020). Equation (7.9) provides the BiGRU network's mathematical properties. The temporal characteristics, which include a variety of information that may be used for classification, are retrieved using the BiGRU network. Yet, these attributes cover a broad range of information that might affect training time, which is the classification layer overhead. Equation (7.6) is applied to generate the attention features from the BiGRU network input features, which are then transferred to the feed forward classification layers through the softmax layer.

$$\boldsymbol{P(F) = GRU\left(\sum_{t=1}^{n}[x_{t,}\, h_{t,} z_{t,} r_t\big(W(t), B(t), \eta(tannh)\big)\right]} \quad \textbf{(7.7)}$$

$$\boldsymbol{P(B) = GRU\left(\sum_{t=1}^{n}[x_{t,}\, h_{t,} z_{t,} r_t\big(W(t), B(t), \eta(tannh)\big)\right]} \quad \textbf{(7.8)}$$

Combining the Equation (7.7) and (7.8)

$$\boldsymbol{P(BiGRU) = P(F) + P(B)} \quad (\mathbf{7.9})$$

Here is a list of combined self-attention (SA) with biGRU feature extraction

$$\boldsymbol{Y = Softmax\ (P\big(BiGRU\big), F(K, Q)\big)} \quad (\mathbf{7.10})$$

7.5.4 Deep Feed Forward Network

DFFNs utilizes multiple layers of neurons where every layer is completely linked to the other behind it. The capacity of DFFNs to get intricate data representations makes them perfect for classification jobs (Goodfellow, I., Bengio, Y. and Courville, A., 2016) Attention Enabled Gated Recurrent Network (AEGRN) integrates attention mechanisms with gated recurrent networks to enhance the learning process. This combination improves detection by making sure the network concentrates on the most suitable elements performance. Feature obtained are fed into the DFFN, where the input layer processes raw features such as flow length, packet size, and other aspects of network traffic.

These features thereafter get passed through the deep feed forward network for categorization that is entirely linked. The feed forward network to be described in detail here – a deep feed forward network or a MLP – also has several layers of nodes. Every node is in direct communication with all nodes in all layers of the same multicast group. This makes the deep feed forward network capable of analyzing various relationships between predictors and the targets besides identify various intricate features in the data.

The features from the BiGRU network are then passing through several hidden layers and each layer is implemented with activation function. Activation functions applicable to deep feed forward network include ReLU, sigmoid function and Tanh function. These functions enable the model to learn other forms of factors that make it more non-linear. The mathematical formula of a DFFN can be given as follows (Cil, A.E et al., 2021):

$$\boldsymbol{h^{(l)} = f(W)^{(l)} h^{(l-1)} + b^{(l)}} \quad (\mathbf{7.11})$$

Where:

- $h^{(l)}$ is the l-th layer's output..
- $W^{(l)}$ s the l − th layer′s weight matrix.

- $b^{(l)}$ is the l − th layer′s bias vector.
- f is the function that activates.

A softmax layer receives the output from the last hidden layer to provide conditional probabilities for every class. The definition of the softmax function is:

$$\boldsymbol{Y = softmax(W)^{(L)} h^{(L-1)} + b^{(L)}} \qquad (\mathbf{7.12})$$

Where:

- $W^{(L)}$ and $b^{(l)}$ final layer weight matrix and bias vector, correspondingly.
- $h^{(L-1)}$ is the final hidden layer output.

The deep feed forward circuit is trained through the back propagation of errors, together with a steepest descent technique in order to reduce the cross-entropy amongst the real class labels and the output class probabilities. The loss of cross-entropy is defined as:

$$\boldsymbol{L = -\sum_{i=1}^{C} y_i log(Y_i)} \qquad \mathbf{(7.13)}$$

Where:

- C is the class count.
- $y_i \; is$ the real class label (i).
- Y_i is class probability prediction (i).

In summary, the features extracted from the BiGRU network are input into a deep feed forward network, this includes a final softmax layer for classification along with many hidden layers. This method makes use of the deep feed forward network's classification skills and the BiGRU network's potent feature extraction capacity to achieve high performance in the classification task.

The Table 7.1 shows pseudocode for proposed GRU-Based Feature Extraction and Classification with Integrated Self-Attention Mechanism

Table 7.1 Pseudocode for proposed GRU-Based Feature Extraction and Classification with Integrated Self-Attention Mechanism

```
Step 1: # GRU Equations
function gru(input_xt, prev_ht, Wh, Uh, bh, Wr, Ur, br):
    # Update Gate
    z_t = sigmoid(dot_product(Wh, input_xt) + dot_product(Uh, prev_ht) + bh)
    # Reset Gate
    r_t = sigmoid(dot_product(Wr, input_xt) + dot_product(Ur, prev_ht) + br)
    # Candidate Activation
    h_t_tilde = tanh(dot_product(Wh, input_xt) + dot_product(Uh, r_t * prev_ht) + bh)
    # Update the Hidden State
    h_t = (1 - z_t) * prev_ht + z_t * h_t_tilde
    return h_t

Step 2: # Self-Attention Equation
function self_attention(K, Q, V):
    # Attention Calculation
    attention_scores = dot_product(K, Q^T) / sqrt(K.shape[0])
    # Softmax Attention Weights
    attention_weights = softmax(attention_scores)
    # Weighted Sum of Values
    attention_output = dot_product(attention_weights, V)
    return attention_output

Step 3: # BiGRU Feature Extraction
function bigru_feature_extraction(input_xt, prev_ht, Wh, Uh, bh, Wr, Ur, br):
    # Forward Pass
    forward_ht = gru(input_xt, prev_ht, Wh, Uh, bh, Wr, Ur, br)
    # Backward Pass
    backward_ht = gru(reverse(input_xt), reverse(prev_ht), Wh, Uh, bh, Wr, Ur, br)
    # Concatenation of Forward and Backward Hidden States
```

```
    bigru_output = concatenate(forward_ht, backward_ht)
    return bigru_output
Step 4: # Integrated Self-Attention with BiGRU Feature Extraction
function integrated_attention_bigru(input_xt, prev_ht, K, Q, V, Wh, Uh, bh, Wr, Ur, br):
    # BiGRU Feature Extraction
    bigru_output = bigru_feature_extraction(input_xt, prev_ht, Wh, Uh, bh, Wr, Ur, br)
    # Self-Attention
    attention_output = self_attention(K, Q, bigru_output)
    return attention_output
Step 5: # Feed Forward Classification Layers
function feed_forward_classification(input_features, weights):
    # Fully Connected Layer
    hidden_layer = dot_product(input_features, weights)
    # Softmax Activation
    output = softmax(hidden_layer)
    return output
```

7.6 Experiment Details

An Intel Workspace with a frequency of 3.2 GHz, an I7 CPU and 16GB of RAM is used to construct the complete algorithm. The proposed basic infrastructure is developed using Keras (integrated into Tensorflow) as the back end.

7.7 Performance Metrics

For the purpose to validate the suggested structure and experiments are being carried out to develop deep feed forward training networks to correctly categorize regular sensitive and attack data. The performance metrics are the characteristics used to evaluate the effectiveness of the suggested design and the experimental hyperparameters used to train the proposed network shown in Table 7.2.

Table 7.2 Hyper parameters employed in the proposed network's training

S.No	Hyper-Parameter	Specifications
1	Number of GRU Cells	10
2	Number of Epochs	250
3	Number of Batch Size	30
4	Learning Rate	0.001
5	Momentum	0.2
6	Dropouts	0.2

7.8 Results and Discussion

Component architectures that have identical parameters as the suggested system are utilized for the experiments. One-dimensional LSTM (Staudemeyer RC, 2015), Gated Recurrent Units (Kim J et al., 2016), Optimized LSTM (Kimmel J.C et al., 2021), and BiGRU (Shen Y et al., 2018) are specific examples of the present architectures. Four distinct datasets are used in an evaluation once the approach has been verified.

The classification models validated in this chapter are,

(i) LSTM

(ii) GRU

(iii) Optimized -GRU

(iv) Proposed Attention Enabled Gated Recurrent Network - Deep Feed Forward Network – (Proposed AEGRN-DFFN)

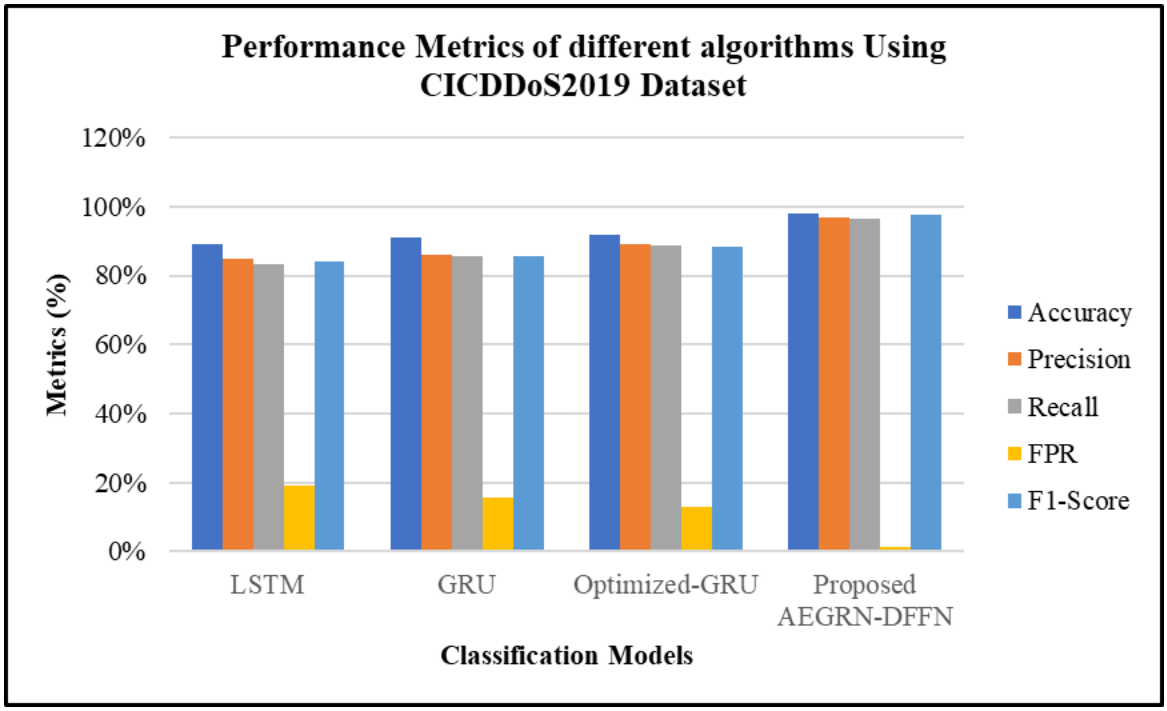


Figure 7.3 Performance Comparison using CICDDoS2019 Dataset

Table 7.3 Performance metric values for CICDDoS2019 Dataset

Techniques	Performance Metrics				
	Accuracy (%)	Precision (%)	Recall (%)	Specificity (%)	F1-Score (%)
LSTM	89.00	85.00	83.40	19.00	84.00
GRU	91.00	86.00	85.60	15.50	85.70
Optimized-GRU	92.00	89.00	88.70	12.90	88.50
Proposed AEGRN-DFFN	**98.00**	**97.00**	**96.60**	**1.10**	**97.50**

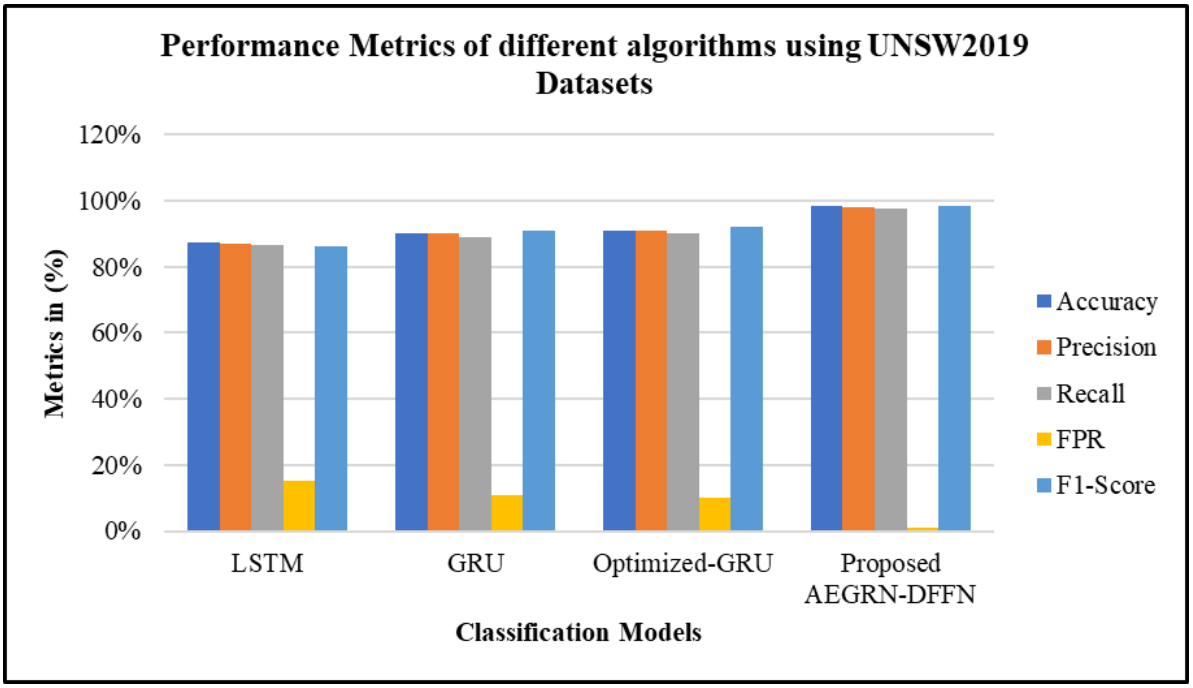


Figure 7.4 Comparing Performance with the UNSW2019 Dataset

Table 7.4 Performance metric values for UNSW2019 Dataset

Techniques	Performance Metrics				
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**Specificity (%)**	**F1-Score (%)**
LSTM	87.40	87.00	86.40	15.00	86.00
GRU	90.20	90.00	89.00	11.00	91.00
Optimized-GRU	91.00	91.00	90.00	10.00	92.00
Proposed AEGRN-DFFN	**98.30**	**98.00**	**97.40**	**1.10**	**98.30**

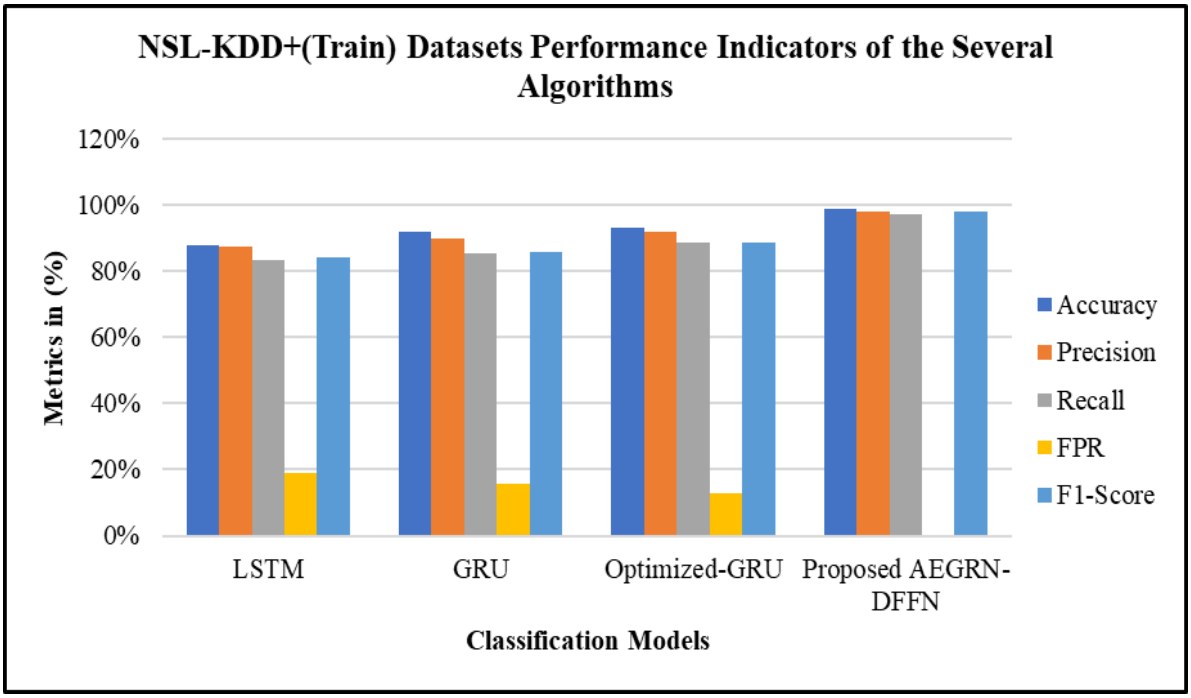


Figure 7.5 Performance Comparison using NSL-KDD+(Train) Dataset

Table 7.5 Performance metric values for NSL-KDD+(Train) Dataset

Techniques	Performance Metrics				
	Accuracy (%)	Precision (%)	Recall (%)	Specificity (%)	F1-Score (%)
LSTM	88.00	87.50	83.40	19.00	84.00
GRU	92.00	90.00	85.60	15.56	85.70
Optimized-GRU	93.00	92.00	88.70	12.90	88.50
Proposed AEGRN-DFFN	**98.80**	**98.00**	**97.40**	**0.10**	**98.00**

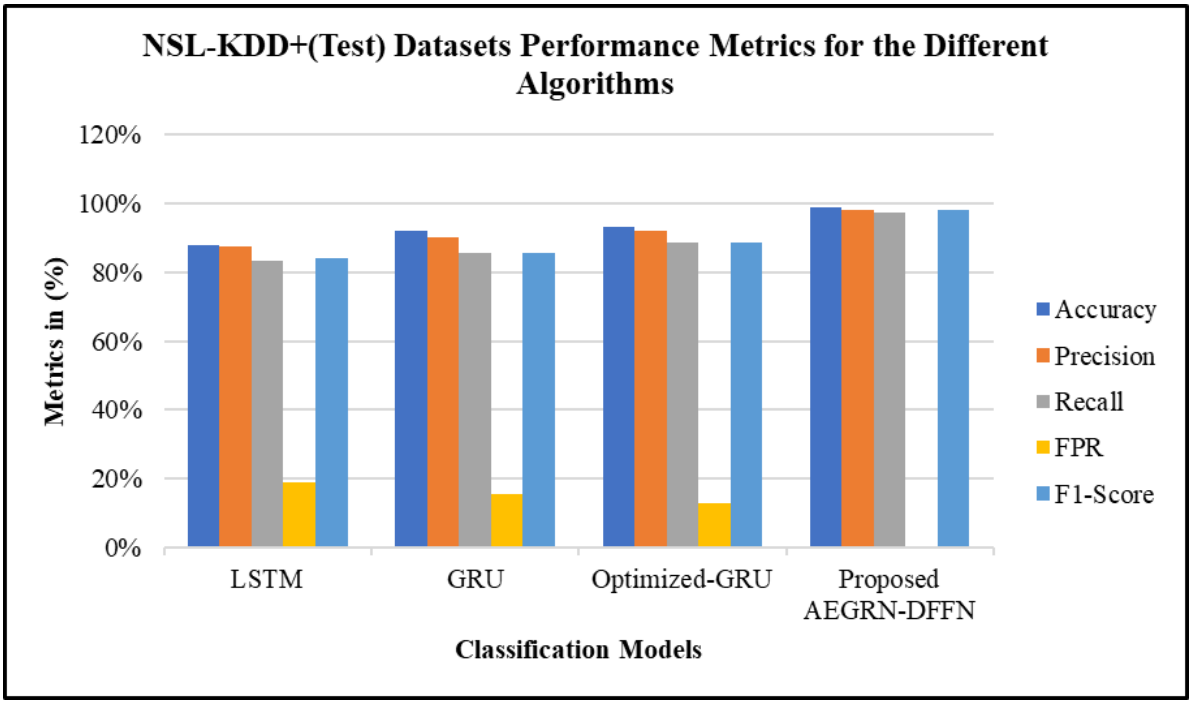
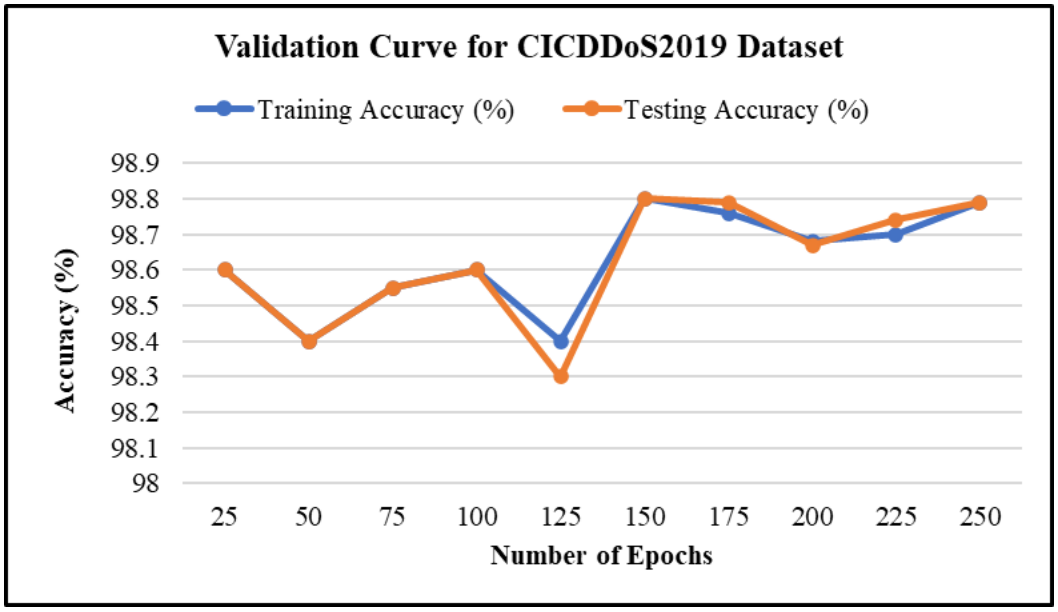


Figure 7.6 Performance Comparison using NSL-KDD+(Test) Dataset

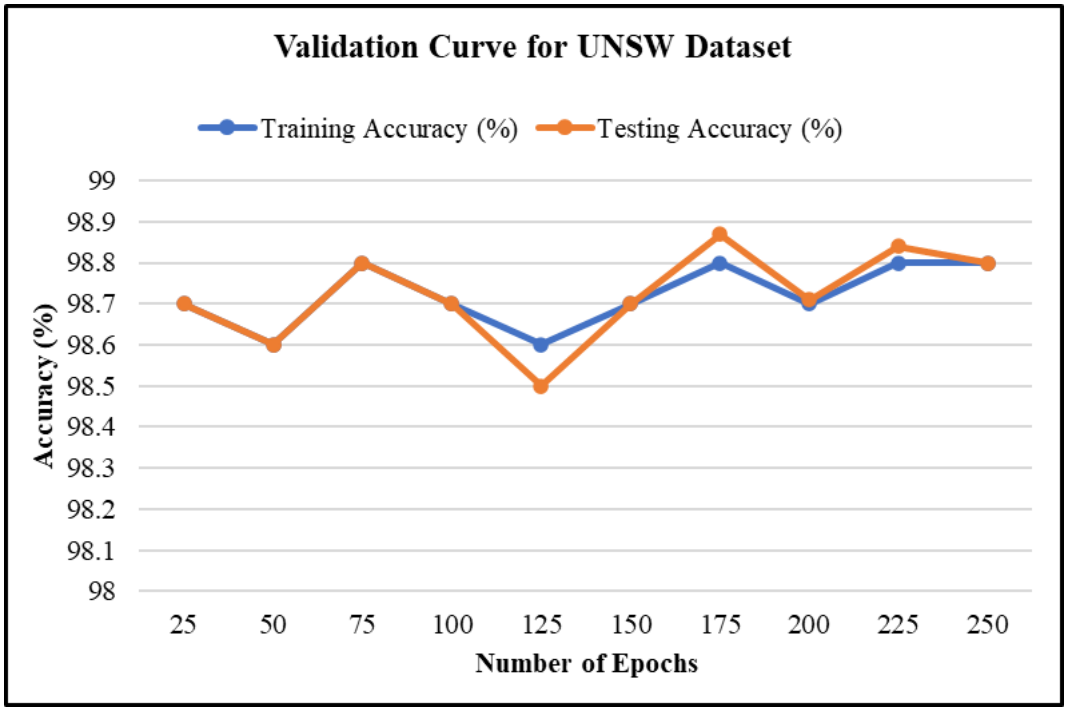
Table 7.6 Performance metric values for NSL-KDD+(Test) Dataset

Techniques	Performance Metrics				
	Accuracy (%)	**Precision (%)**	**Recall (%)**	**Specificity (%)**	**F1-Score (%)**
LSTM	88.00	87.50	83.40	19.00	84.00
GRU	92.00	90.00	85.60	15.56	85.70
Optimized-GRU	93.00	92.00	88.70	12.90	88.50
Proposed AEGRN-DFFN	**98.80**	**98.00**	**97.40**	**0.10**	**98.00**

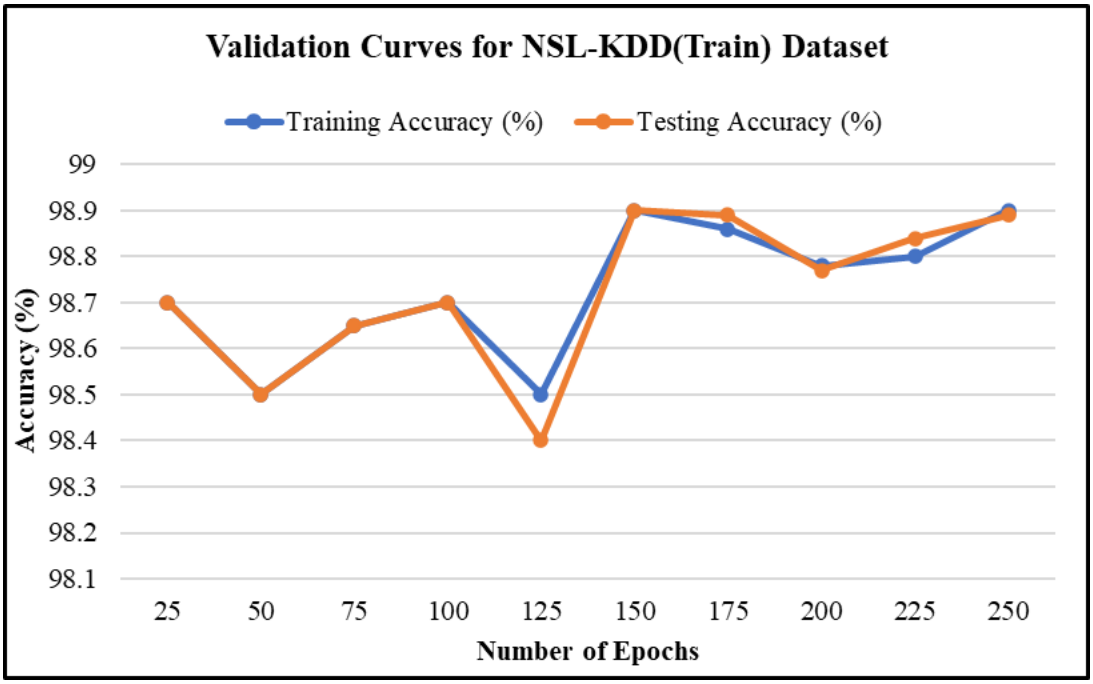
Additionally, a range of datasets are utilized to access the proposed model's validation efficacy, and it is demonstrated that the RMSE (root mean square error) among training and testing data is 0.001.



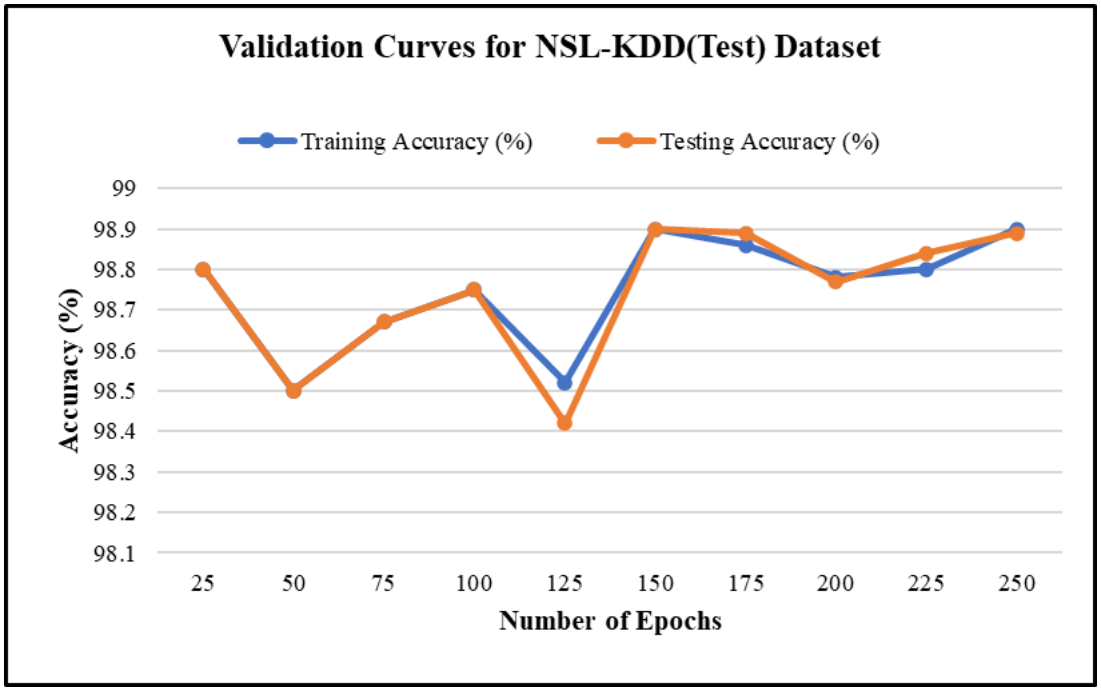
(a) Validation Curves for CICDDoS2019 Dataset with Number of Epochs



(b) Validation Curves for UNSW Dataset with Number of Epochs



(c) Validation Curves for NSL-KDD(Train) Dataset with Number of Epochs



(d) Validation Curves for NSL-KDD(Test) Dataset with Number of Epochs

Figure 7.7 Validation Performance of the suggested Model using distinctive datasets a) CICDDoS2019 datasets b) UNSW-datasets c) NSL-KDD datasets (Train) d) NSL-KDD datasets (Test)

Figure 7.3 to 7.6 and Tables 7.2 to 7.5 demonstrate performance metrics of the proposed method for classifying multiple attacks using different datasets. When it comes to identifying the various assaults, the suggested model AEGRU-DFFN has displayed the best performance. The application of self-attention maps provides the greatest outcomes when compared to numerous other DL-based techniques. Additionally, validation effectiveness of proposed model (Figure 7.7) is assessed across a range of datasets, providing an RMSE of 0.001 among the test and training data.

Table 7.7 Comparison of Various Algorithms with Various Datasets on Model Built Time

Datasets	(MBT)-secs (per epoch)			
	LSTM	**GRU**	**Op-LSTM**	**Proposed Model**
CICDDoS2019	34.2	31.2	26.4	**18**
UNSW	31.2	27.6	22.8	**16.8**
NSL-KDD++ Train	34.2	31.2	26.4	**17.4**
NSL-KDD++ Test	30	29.4	27	**17.4**
Average MBT	32.4	29.85	25.65	**17.4**

Table 7.7 displays model construction timeframes for different classifiers for four datasets that use hold-out assessment. The realization that it is critical to take into account the duration a model requires training before it is successfully spotting various threats is the main driving force for calculating MBT (De Sousa M.S et al., 2022) (Alabdulwahab, Saleh, and Bong Kyo Moon 2020). As a result, MBT helps to attain a better balance between classifier performance and computational complexity. As per the above table, the average MBT for training different datasets is 17.4 s per epoch, whereas Op-LSTM, GRU, and LSTM each use 32.4 s, 29.85 s, and 25.65 s, respectively. The research concludes that the proposed model finds a solid role in designing the defensive system against various assaults and uses just 17.4 seconds.

The performance metrics of the proposed model (AEGRN-DFFN) underscore its remarkable efficiency and robustness, thereby justifying its scalability and reduced

computational complexity and overhead. The model exhibits high accuracy (98.00% to 98.80%), precision (97.00% to 98.00%), recall (96.60% to 97.40%), and F1-Score (97.50% to 98.30%) across various datasets, indicating its capability to consistently identify both normal and attack traffic with minimal errors. These metrics suggest that the model can handle varying data distributions effectively, making it appropriate for use in more expansive and varied settings without significant performance degradation. The exceptionally low model building time (MBT) of 17.4 seconds per epoch, compared to other models, highlights its efficient training process, which translates to lower computational overhead. Although the model's specificity values are low (0.10% to 1.10%), this trade-off prioritizes detecting true positives over avoiding false positives, ensuring comprehensive threat detection. The high precision and recall mitigate the impact of low specificity, maintaining overall detection effectiveness and minimizing the computational burden associated with false positives. These attributes collectively validate the model scalability and efficiency, confirming that it is appropriate for real-world applications.

7.9 Chapter Summary

This chapter explores the combination of self-attention maps with GRU to safeguard the network against various threats. To select the best attributes that may facilitate the classification layers, this study recommends combining the BiGRU with a self-attention network. Additionally, in order to accomplish better classification with less computing load and faster speed, the suggested study has made use of feed forward layers that operate on the idea of Extreme Learning Machines. Every classifier is benchmarked using the CICDDoS2019, UNSW-NB15, and NSL-KDD datasets. The outputs show that the suggested method surpasses the other deep learning methods with respect to higher detection ratio and less overhead. This approach is also complexity-aware, adapting to different data complexities to provide accurate and efficient classification while managing the complexities of real-time assault patterns. Future scope: The efficacy of the suggested approach is necessary for both brighter deployment under resource constraints and validation using real-time datasets.