DHPAN: A Real-Time Adaptive Deep Learning Framework for Streaming Intrusion Detection

Balaji D
Ph.D Scholar
Indian Institute of Industry Interaction Education and Research (IIIIER) Chennai, India

Abstract — The growing number and complexity of network traffic in contemporary cloud and IoT systems require dynamic and real-time systems of intrusion detection. In this paper, a multimodal deep learning framework known as Dynamic Hyper-Parallel Attention Network (DHPAN) is presented, which is used to detect intrusion by streaming. The DHPAN combines a Lightweight Feature Embedding Layer (LFEL) to support compact representations, a Convolutional Recurrent Neural Network (CRNN) with Anomaly-Aware Attention (AAA) to both learn spatial-temporal and contextual patterns and a Hyper-Parallel Dual Optimization (HPDO) strategy that incorporates both Adaptive Adam and Evolutionary RMSprop. The framework applies seven stages of processing network traffic, such as preprocessing and adaptive feature selection, interpretability, and evaluation. The experimental findings on NSL-KDD and CIC-IDS-2017 data sets show a higher performance, with accuracy of 99.59% and 99.68%, respectively. Measures made at network level show low detection latency, high throughput and low packet drop, which confirm real-time applicability. Ablation and threshold sensitivity tests validate the role of every module and the most suitable selection of anomaly score. DHPAN provides proactive zero-day and multi-vector attack response when preserving network QoS, which provides a high-quality, scalable and understandable intrusion detection platform on next-generation networked systems.

Keywords: Anomaly Detection, Feature Embedding, Intrusion Detection, Network Traffic, Parallel Attention

I. INTRODUCTION

The advanced infrastructures have become extremely susceptible to sophisticated cyber-attacks, including Denial-of-Service (DoS), distributed multi-vector, and zero-day attacks, due to the exponential growth of the volume of network traffic, its velocity, and heterogeneity as a result of the rapid development of cloud computing [1] and Internet of Things (IoT) ecosystems [2]. Such attacks pose a serious risk to the availability of systems, the confidentiality of data, and the integrity of services in cloud-IoT systems [3], [4]. Conventional intrusion detection systems (IDS) which mostly use signature-based detection cannot be used to detect new and emerging attacks because they lack the capability to generalize beyond the known attacks [5-7].

In order to address these limitations, a number of Machine Learning (ML)- and Deep Learning (DL)-based IDS solutions have been suggested to be used in IoT [8-10] and cloud settings [11]. Existing classifiers like Decision Trees, the Support Vector Machines (SVM), and ensemble models enhance the detection rate but have scalability and false positive rates issues when they are subjected to heterogeneous traffic streams [12]. The deep learning models such as CNNs, RNNs, and hybrid models have proven to be more effective in feature learning because they can learn on a spatial or temporal dependency [13]. Nevertheless, the current models do not model complex spatial-temporal relationships, dynamic attack behavior adaptive methods and perform effectively in circumstances of real-time, high throughput [14], [15].

Additionally, it is observed that most of the state-of-the-art IDS models have high computation overheads, slow convergence, and lack expressiveness to zero-day attacks, which directly affect network Quality of Service (QoS) parameters as latency, throughput, and ratio of packet delivery [16]. These issues also demonstrate a significant research gap in creating an IDS that is adaptive, low-latency, highly accurate, and resistant to various attack conditions and retains operational efficiency in cloud-IoT systems [17].

In response to this, this paper suggests the use of a multimodal deep learning model called a Dynamic Hyper-Parallel Attention Network (DHPAN) which combines a Lightweight Feature Embedding Layer (LFEL), a Convolutional Recurrent Neural Network (CRNN) with Anomaly-Aware Attention (AAA) and Hyper-Parallel Dual Optimization (HPDO). The suggested architecture already manages the spatial-temporal dependencies in the most efficient way and focuses on anomalous traffic patterns and guarantees rapid and consistent convergence when dealing with large volumes of streaming data.

- The most important works of such work can be summarized as follows:
- An adaptive framework of real time intrusion detection with a high accuracy on a variety of attack types, including zero-day attacks.
- Available features a LFEL to minimize computational costs at the cost of performance in detection.
- CRNN with anomaly-aware attention: Enhanced spatial-temporal anomaly detection.
- A hyper-parallel dual optimization strategy that allows the efficient training and rapid convergence.
- Broad assessment of benchmark datasets (NSL-KDD and CIC-IDS-2017) in terms of both detection measures and network-level QoS.

II. BACKGROUND STUDY

The most recent studies have been aimed at using deep learning, ensemble learning, SDN, and blockchain methods to augment the intrusion detection systems of IoT and cloud-enabled systems. Such studies were designed to enhance accuracy of detection, versatility and scalability in challenging attack conditions. Nevertheless, the issues concerning the computational overhead, real-time processing, and the detection of a zero-day attack are not solved yet.

Lazzarini et al. (2023) [18] suggested a deep learning IDS algorithm based on stacking ensemble in the application to IoT networks, which would increase detection strength. A meta-classifier was used to combine multiple deep models to enhance the generalization across the types of

attacks. The method performed well on the benchmark sets. Nevertheless, the ensemble architecture raised the level of computation and reduced the applicability to real time and resource constrained IoT settings.

Res-TranBiLSTM introduced by Wang et al. (2023) [19] is a variant of BiLSTM that incorporates residual connections with Transformer mechanisms and is used to apply long-term temporal dependencies in IoT traffic. The model had better detection and stability than the conventional RNN-based IDS. However, the architecture was costly in terms of training and memory usage which limited scalability in large-scale deployments.

Gaber et al. (2023) [20] created a deep learning intrusion detection framework, named Metaverse-IDS, designed to work with IoT networks based on the Metaverse. The system was able to identify complex attacks patterns and it was able to classify at high rates. Although it was effective, the framework was not lightweight optimized, which is not suitable in edge and low-power IoT devices.

A deep learning-based IDS was suggested by Chaganti et al. (2023) [21] to monitor traffic and detecting attacks centrally in the IoT network, along with Software Defined Networking (SDN). The model enhanced efficiency in detection by the visibility of the global network. Nonetheless, the use of a centralized SDN controller created a latency and single-point-of-failure problem.

Wardana et al. (2024) [22] introduced a collaborative IDS that was designed to be lightweight, trust-management based and privacy-preserving in IoT environments. The framework minimized the calculational costs at the expense of decent accuracy in detection. However, its lightness did not allow it to be effective in identifying advanced and zero-day attacks.

The lightweight supervised IDS mechanism was suggested by Roy et al. (2022) [23] and emphasized the reduction of resources usage in the IoT networks. The method obtained a satisfactory detection performance at a low processing cost. The system was however based on predetermined attack patterns, and this minimized flexibility to emerging and unknown threats.

Alshahrani et al. (2023) [24] created an SDN-based intrusion detection model of Industrial IoT setting. The model increased the accuracy of attack detection using programmable network control. Although quite effective, the framework had the disadvantage of more control plane overhead and reduced real time responsiveness.

Sharadqh et al. (2023) [25] proposed a graph-based attack mitigation blockchain-enabled hybrid IDS framework on edge-assisted IoT settings. The system enhanced trust, non-secrecy, and response to intrusion. Nonetheless, the use of blockchain presented both computational and communication overheads, which affected latency-sensitive IoT applications.

Ref.	Method / Framework	Core Technique	Key Results	Limitations
Lazzarini et al. (2023) [18]	Stacking Ensemble IDS	Ensemble deep learning with meta-classifier	Achieved high detection accuracy and improved generalization	High computational complexity; limited real-time suitability
Wang et al. (2023) [19]	Res-TranBiLSTM	Residual learning + Transformer + BiLSTM	Improved temporal feature learning and detection performance	High training cost and memory overhead
Gaber et al. (2023) [20]	Metaverse-IDS	Deep learning for Metaverse-IoT	High accuracy in complex IoT attack scenarios	Not optimized for resource-constrained devices
Chaganti et al. (2023) [21]	SDN-enabled DL IDS	Deep learning with centralized SDN control	Enhanced global traffic visibility and detection	Latency and single-point-of-failure risk
Wardana et al. (2024) [22]	Lightweight Trust-based IDS	Collaborative, privacy-preserving IDS	Reduced overhead with acceptable accuracy	Limited effectiveness against zero-day attacks
Roy et al. (2022) [23]	Lightweight Supervised IDS	Low-complexity supervised learning	Low resource consumption and reasonable detection	Poor adaptability to evolving attacks
Alshahrani et al. [24]	SDN-based IIoT IDS	Programmable SDN-based detection	Improved detection in industrial IoT	Increased control-plane overhead
Sharadqh et al. (2023) [25]	Blockchain-enabled Hybrid IDS	Blockchain + graph-based mitigation	Improved trust and attack mitigation	High computation and communication overhead

Table 1: Comparison Table of Ids Methods.

Table 1 shows the various researches on IDS detection methods used in cloud and IoT environments.

III. PROPOSED METHODOLOGIES

The Dynamic Hyper-Parallel Attention Network (DHPAN) is a multimodal deep learning architecture of real-time intrusion detection in dynamic cloud and IoT networks. It incorporates preprocessing, adaptive feature selection and Lightweight Feature Embedding Layer (LFEL) to more effectively prepare network data with a low level of computation overhead. The framework uses a CRNN backbone with Anomaly-Aware Attention (AAA) modules to draw in the spatial and temporal patterns in traffic and concentrate on the anomalous actions. A Hyper-Parallel Dual Optimization (HPDO) strategy that is a combination of Adaptive Adam and Evolutionary RMSprop is used to accelerate and stabilize the training. The seven-stage architecture provides end-to-end processing, including the ingestion of raw data, all the way to

explainable anomaly detection, to facilitate the rapid, accurate, and interpretable detection of threats.

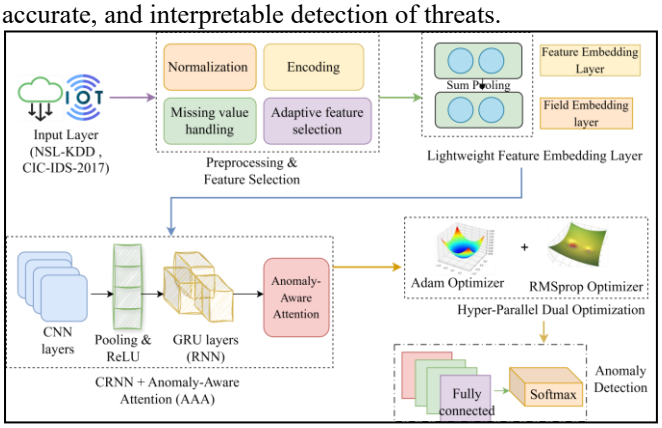

Fig. 1: Hyper-Parallel Attention–Driven IDS Architecture

Figure 1 is based on sequential and end-to-end pipeline of real time intrusion detection. LFEL is used to preprocess network traffic and compress it into compact representations, and a CRNN with Anomaly-Aware Attention is used to normalize both spatial and temporal aspects as well as anomalies. Lastly, Hyper-Parallel Dual Optimization provides stable and efficient learning, thus allowing correct and low-latency detection of anomalies in a wide range of network conditions.

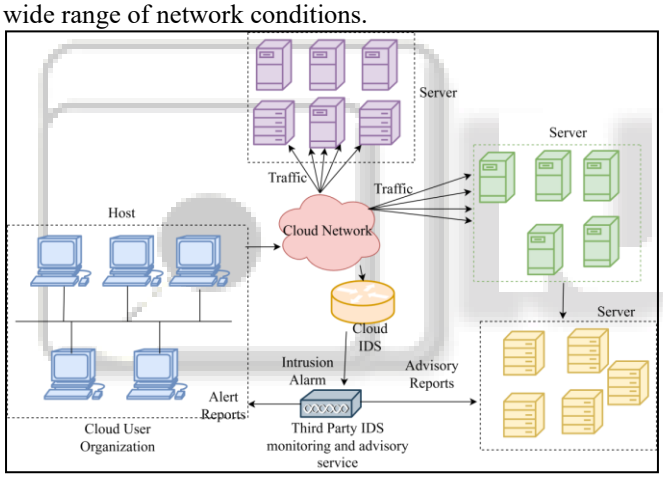

Fig. 2: Network traffic and IDS in Cloud

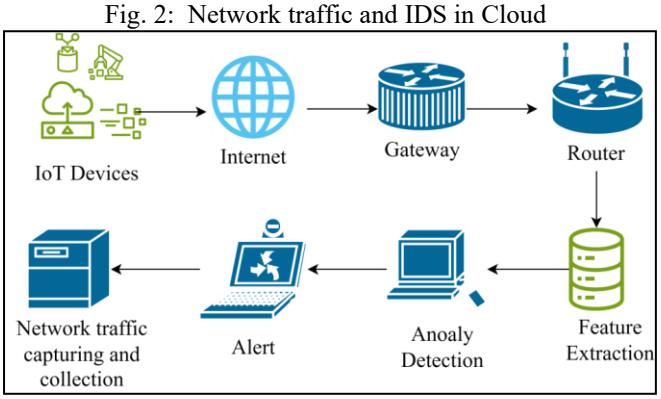

Fig. 3: Network traffic and IDS in IoT

Figure 2 and 3 are used to depict network traffic created by cloud and IoT environment through interconnected devices and gateways to a centralized intrusion detection system. Real time monitoring is done over legitimate and malicious traffic, the IDS processes the patterns to detect intrusions and attacks. This diagram shows how DHPAN ensures heterogeneous cloud-IoT networks through constant monitoring of the traffic and stopping cyber threats.

A. Dataset description

Dataset: https://www.kaggle.com/datasets/hassan06/nslkdd
NSL-KDD dataset is an improvement of original KDD Cup 1999 intrusion dataset which has been made bias and imbalanced due to redundant and duplicate records, and the original feature structure is maintained. It has hundreds of thousands of labeled connection records with 41 features, which denote the normal or specific attack behaviors of DoS, Probe, R2L, and U2R. NSL-KDD is popular as a benchmark in the assessment of intrusion detection systems since it offers a more realistic and balanced assessment environment as compared to its predecessor.

Dataset 2:
https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset

Network-Intrusion-Dataset, which is built on the CIC-IDS-2017 benchmark, includes millions of network traffic logs over several days of benign and malicious traffic, with a wide variety of attack types representative of modern attack types being Brute Force, DoS, DDoS, Heartbleed, and Botnet. It has detailed flow based characteristics that are derived out of actual traffic with supervised labels to be learned, which can be used to train and evaluate machine learning-based intrusion detection model. The purpose of CIC-IDS-2017 is to replicate real world network characteristics including various protocols and user actions.

B. Preprocessing

DHPAN uses preprocessing to clean and prepare raw network traffic using benchmark traffic data like NSL-KDD and CIC-IDS-2017. This can be carried out by eliminating noise and irrelevant entries like duplicates, unutilized fields to minimize redundancy of data. Values are imputed in case of missing or corruption to have all the features completes to analyze them. Continuous variables, e.g. packet length or flow duration are normalized or scaled to have consistent scales, enhancing model stability and convergence. Categorical characteristics, especially the type of the protocol (TCP, UDP, and ICMP) and flags are represented by their numbers or by the one-hot encoding in order to make them processed by the model. Basic feature cleaning and alignment is also part of preprocessing to ensure that the structure of the two datasets is the same. At the end of this stage, the network traffic information is organized, predictable and prepared to utilize adaptive features selection and lightweight feature embedding, which permits effective and precise intrusion detection during the next phases.

$$x'_{j,i} = \frac{x_{j,i} - x_i^{min}}{x_i^{max} - x_i^{min}} \quad (1)$$

Where, $x_{j,i}$ is original value of feature i in sample j, x_i^{min}, x_i^{max} are minimum and maximum of feature i, $x'_{j,i}$ is normalized value. Equation (1) is min max scaling, which scales the feature i of sample j to [0,1] to make them the same so that they are numerically stable and contribute equally in training.

$$z_{j,i} = \frac{x_{j,i} - \mu_i}{\sigma_i} \quad (2)$$

Where, μ_i is the mean of feature i, σ_i is the standard deviation of feature i, $z_{j,i}$ is standardized feature. Equation (2) is Z-score standardization, it makes the mean of the feature i to be equal to zero with unit variance, which enhances convergence of the model and also the variable features that have different ranges.

$$x_{j,i}^{imputed} = \frac{1}{N_i} \sum_{k=1}^{N_i} x_{k,i} \quad (3)$$

Where N_i is number of non-observations values of feature i, $x_{k,i}$ is observed value of feature i.In equation (3), the missing data of feature i are substituted by an average of observed values in order to preserve the integrity of data and prevent errors during training.

$$x'_{j,i,c} = \begin{cases} 1, & \text{if sample j belongs to category c} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Where, $x'_{j,i,c}$ is the value of feature i, category c, sample j, which is coded. Categorical features such as protocol and flags are coded in equation (4) using numbers, and the deep learning model can use them to compute discrete attributes.

C. Adaptive Feature Selection

Following the preprocessing, DHPAN uses adaptive feature selection in order to select and store the most informative and discriminative features of the network traffic data. The process lets dimensionality be reduced, removes redundant or insignificant attributes and enhances the training efficiency and generalization of the model. The features are considered depending on their statistical significant or relevance score and only those which add the most to anomaly detection are picked. With such high-impact features, DHPAN makes sure that the following embedding and CRNN modules work on a smaller meaningful feature set, higher detection rates with less computation cost.

$$S(f_i) = \frac{1}{N} \sum_{j=1}^{N} | x_{j,i} - \mu_i | \quad (5)$$

which, $x_{j,i}$ is the value of feature i in sample j, μ_i is the mean of feature i, $S(f_i)$ is the importance score of feature i. To measure the relevance of a feature, the average deviation of feature iii is computed in Equation (5) and features that are most indicative of anomalies are retained.

$$f_i^{selected} = \begin{cases} f_i, & \text{if } S(f_i) \geq \tau_s \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

Where τ_s is selection threshold, $f_i^{selected}$ is selected feature. Equation (6) is used to filter redundant or irrelevant features by choosing feature i whose importance score is greater than threshold τ_s.

$$X_s = [f_1^{selected}, f_2^{selected}, \dots, f_{F_s}^{selected}] \quad (7)$$

In which, $X_s \in \mathbb{R}^{N \times F_s}$ is feature matrix post-selection, F_s is selected features. Equation (7) generates the matrix of retained features, which supplies a concise significant input to the embedding layer.

D. Feature Embedding

Once the adaptive features are selected, DHPAN submits the features selected to the Lightweight Feature Embedding Layer (LFEL). LFEL has the ability of projecting high-dimensional network features into low-dimensional embeddings that maintain the important information but makes the computation complex low. The step allows the following CRNN backbone to work effectively with the data without losing important patterns relating to anomalies. LFEL accelerates training and inference by placing features into less memory, and makes it applicable in high-throughput network settings in real-time, and less memory usage. In essence, LFEL is a feature compressor and translator and therefore the deep learning model is lightweight and effective in intrusion detection.

$$E = X_s W + b \quad (8)$$

Which, $X_s \in \mathbb{R}^{N \times F_s}$ is the choice of features, $W \in \mathbb{R}^{F_s \times d}$ is weight matrix, $b \in \mathbb{R}^{1 \times d}$ is bias, $E \in \mathbb{R}^{N \times d}$ is embedded features. Equation (8) maps the chosen features into a reduced dimensional array E, which does not lose any important information but saves computation costs.

$$E' = \mathrm{ReLU}(E) \quad (9)$$

Where, E' is an activated embedded feature. The equation (9) uses non-linearity on embeddings to learn complex interactions of features, and improve the learning of representations.

E. Pattern Extraction Using CRNN with Anomaly-Aware Attention

Once the feature embedding occurs with LFEL, the scaled representations are inputted into the Convolutional Recurrent Neural Network (CRNN) backbone with Anomaly-Aware Attention (AAA) that is the gist detection engine of DHPAN. The convolutional layers are subsequently applied to detect the spatial patterns of each traffic sample learning about local feature interactions, including correlations of packet size, duration, protocol type, and flow statistics, revealing the detection of thin slices of anomalous relations. The layers are repeated, normally by GRU units, and then learn temporal relationships across sequence of traffic logs, which is basic to detecting time-varying attacks, such as denial-of-service floods and scanning patterns. In order to improve the detection power, the AAA module dynamically reduces attention proportion on the time steps and dimension of features that are characterized as anomalous enabling the model to concentrate on the attack-relevant patterns and exclude benign traffic noise. Consequently, this step is able to produce context-sensitive representations that combine the spatial, temporal, and anomaly-centric data that are further employed to classify the intrusions accurately or score the anomalies.

$$H_c = \mathrm{ReLU}(E' * W_c + b_c) \quad (10)$$

Where, E' is LFEL output, W_c is convolution filters, b_c is convolution bias, H_c is spatial feature map. Equation (10) obtains spatial correlations between embedded features which brings out local patterns that represent anomalies.

$$H_p = \max(H_c^{(window)}) \quad (11)$$

Where $H_c^{(window)}$ is local receptive field, H_p is pooled features. Equation (11) singularly lowers the feature map dimensionality and preserves the largest worthwhile local spatial patterns to be effectively processed.

$$H_c(i) = \sum_{j=1}^{K} E'(i + j - 1) \cdot W_c(j) + b_c \quad (12)$$

Where, K is the size of the kernel, used in the convolutional layer, j is the index of the kernel, b_c is the convolutional bias term, $H_c(i)$ is the raw convolutional feature response at point i. The convolution operation is explicitly expanded in equation (12) to demonstrate the

manner in which the local neighborhoods of embedded features are summed up to explain how spatial interactions of features play a part in extracting anomaly patterns.

$$H_c^1 = \max(0, H_c) \quad (13)$$

Where, H_c^1 is the activated spatial feature map. In this equation (13), the ReLU activation is used to eliminate negative responses and only salient spatial activations are retained that are cues to abnormal traffic behavior.

$$H_r = \text{RNN}(H_p) \quad (12)$$

Where, H_p is pooled spatial features, $H_r \in \mathbb{R}^{N\times T\times h}$ is hidden states at timesteps, T is sequence length, h is hidden units. Equation (12) models sequential dependencies of temporal-dependent traffic and represents patterns over time that are important in identifying attacks.

$$zt = \sigma(W_z H_p(t) + U_z ht - 1 + b_z) \quad (13)$$

Where, zt is Update gate vector step t, $H_p(t)$ is pooled spatial feature vector step t, $ht - 1$ is previous hidden state, ht is the final hidden state, W_z is input weight, U_z is recurrent weight, b_z is bias vector. The update gate is calculated in equation (13) to determine the extent to which the past traffic information would be retained during the timestep being processed.

$$rt = \sigma(W_r H_p(t) + U_r ht - 1 + b_r) \quad (14)$$

Where, rt is the reset gate vector at time step t, W_r is the input weight, U_r is the recurrent weight, b_r is the bias vector. Equation (14) will decide the extent of the past time context that the model ought to disregard so that it can quickly adapt to the abrupt changes in the traffic.

$$\tilde{h}t = tanh(W_h Hp(t) + U_h(rt \odot ht - 1) + b_h) \quad (15)$$

Where, $\tilde{h}t$ is candidate hidden state, W_h is input weight, U_h is recurrent weight, b_h is bias vector. The resulting candidate temporal representation created in equation (15) takes into account current spatial features and selectively remembered information in the past.

$$ht = (1 - zt) \odot ht - 1 + zt \odot \tilde{h}t \quad (16)$$

The final GRU hidden state is obtained as a result of equation (16), which adapts a combination of past context with emerging temporal patterns that may be applied to intrusion detection.

$$\alpha_t = \frac{\exp(H_r^{(t)} W_a)}{\sum_{k=1}^{T} \exp(H_r^{(k)} W_a)} \quad (17)$$

Where, $H_r^{(t)}$ is hidden state of RNN at timestep t, W_a attention weight matrix, α_t attention score at timestep t. The relative importance of each timestep is calculated in equation (17) so that the model can target suspicious events in sequences.

$$H_a = \sum_{t=1}^{T} \alpha_t H_r^{(t)} \quad (18)$$

Where, H_a is anomaly-aware embedding. In equation (18), the time characteristics are consolidated based on the attention scores and the patterns that are important to the downstream anomalies detection are highlighted.

$$\sum_{t=1}^{T} \alpha_t = 1 \quad (19)$$

Equation (19) is used to make attention scores a valid probability distribution, which enhances the interpretability and stability of anomaly-aware weighting.

$$H_a = \alpha \odot H_r \quad (20)$$

The element-wise scaling of attention between temporal features (equation (20)) directly models attention variations between features, which contributes to the anomaly-focused representation when performing detection.

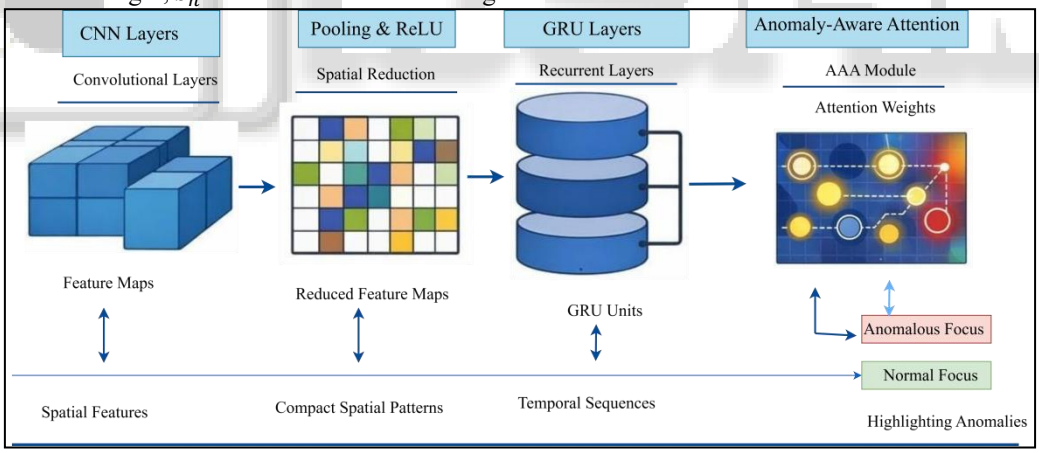

Fig. 4: Pattern Extraction Architecture

Figure 4 demonstrates how the network traffic features are initially represented through the CNN layers to retrieve the spatial correlations, which is then replaced by the pooling and ReLU processes to preserve the major patterns on the local level. GRU layers are then used to model temporal dependence among sequences of traffic, where the AAA module places more attention weight on suspicious time steps, where anomalous behavior is focused. The accuracy and interpretability of intrusion detection is possible due to this spatial-temporal fusion.

F. Optimization (HPDO)

Once the CRNN with AAA has been used to produce anomaly-centric representations, DHPAN utilizes a Hyper-Parallel Dual Optimization (HPDO) approach in order to obtain efficient and stable parameter learning. HPDO implements the Adaptive Adam and Evolutionary RMSprop optimizers concurrently, and the model can enjoy the advantages of both speedy convergence and stability in training. The Adam optimizer adapts parameters based on the first-order moment estimates, which allows rapid learning in high dimensions feature spaces. Simultaneously, RMSprop controls the update of parameters by using a moving average

of squared gradients, which minimize oscillations and enhance stability when operating in non-stationary traffic. This dual-optimizer system is a way of combating the shortcoming of using a single optimalization approach, especially when using heterogeneous and streaming network information. This means that the speed of convergence, throughput and overall performance in generalization are improved by HPDO to allow the detection of various and hitherto unidentified attack patterns.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L_t \quad (21)$$

Where, β_1 is the decay factor, m_t is first moment (mean of gradients), L_t is loss. Equation (21) is used to monitor the exponential gradients averaged to direct the adaptive changes in the Adam optimizer.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta L_t)^2 \quad (22)$$

Where, β_2 is decay factor, v_t is second moment (variance of gradients). Equation (22) computes exponentially weighted variance of gradients to normalize updates and maintain stability.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (23)$$

Equation (23) eliminates initialization bias in Adam moments in order to increase the accuracy of parameter updates.

$$\theta_{t+1}^{(Adam)} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \quad (24)$$

ϵ is a small positive constant added to the denominator for numerical stability. Equation (24) invokes bias-corrected moment-based adaptive gradient descent safely and more quickly.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)(\nabla_\theta L_t)^2 \quad (25)$$

Equation (25) is used to compute the exponential moving average of the squared gradients with weights in order to estimate the more recent size of the parameter changes. Here, $E[g^2]_t$ is the sum of squared gradients at iteration t, $\gamma \in (0,1)$ is the decay term, which decays the memory of previous gradients, $\nabla_\theta L_t$ is the gradient of the loss function L with respect to parameters θ, and the squared gradient is element-wise. This dynamic learning adapts the learning process and makes it stable by eliminating sensitivity to massive gradual shifts.

$$\theta_{t+1}^{(RMS)} = \theta_t - \eta \frac{\nabla_\theta L_t}{\sqrt{E[g^2]_t+\epsilon}} \quad (26)$$

This update rule is the model parameters with normalized gradients with η the learning rate and ϵ is a small constant that is added to improve numerical stability. Adaptive scaling of the step size by the square root of $\sqrt{E[g^2]_t}$ permits RMSprop to converge more quickly and more reliably, particularly in non-stationary or streaming intrusion detection techniques.

$$\theta_{t+1} = \frac{\theta_{t+1}^{(Adam)} + \theta_{t+1}^{(RMS)}}{2} \quad (27)$$

Where, Adam and RMSprop update combinations are combined to take advantage of speed and stability in optimization in Equation (27).

G. Anomaly Detection

DHPAN does the anomaly detection in the aftermath of optimization through HPDO on the anomaly-aware representations generated by CRNN with Anomaly-Aware Attention. These representations represent correlations between spatial features as well as temporal traffic dynamics, and attention is focused on suspicious patterns. A layer of fully connected transforms the learned embeddings into the output score of each of the classes which are then converted to probability through a softmax function to detect multi-class intrusion. In binary cases, the probability of attack is predetermined to the attack to differentiate normal and anomalous traffic in real time. Such a scoring scheme that is threshold-based makes this possible so that rare and zero-day attacks can be effectively detected in streaming situations. Consequently, a score of anomaly is attached to every traffic instance, and a label of detection is presented to facilitate prompt and positive action against network security.

$$y_i = W_o H_a + b_o \quad (28)$$

Where, H_a is anomaly-aware embedding, W_o is output layer weights, b_o is bias, y_i is raw scores for sample i. Equation (28) demonstrates fully connected layer, is maps anomaly-aware embeddings to raw class scores for each sample, forming the basis for classification.

$$p_{i,c} = \frac{\exp(y_{i,c})}{\sum_{k=1}^{C}\exp(y_{i,k})} \quad (29)$$

Where $p_{i,c}$ is the probability of sample i in class c, C is number of classes. Equation (29) transforms raw scores into class probabilities making the outputs readable and the sum of the outputs to 1.

The DHPAN model is taught to reduce the difference between the outputs and ground truth labels by the cross-entropy loss function. This loss allows the model to be able to differentiate normal and anomalous network traffic across various types of attacks.

$$L = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} log(\hat{y}_{i,c}) \quad (30)$$

Where, N is the number of samples in a mini-batch, C is the number of classes (including normal and attack types), $y_{i,c}$ is the ground truth of sample i belonging to class c, and $\hat{y}_{i,c}$ is the predicted probability of belonging to class c. The loss function expressed as equation (30) directs the network to give more probabilities to correct classes and punish the networks where the predictions are incorrect so that the learning of both normal and anomalous patterns can occur.

DHPAN after training calculates an anomaly score on each network traffic sample in order to have a measure of its probability of being malicious. Multi-class outputs are based on the probability of attack classes that are predicted.

$$\text{label}_i = \begin{cases} 1, & \text{if } p_{i,attack} \geq \tau \quad \text{(anomalous)} \\ 0, & \text{otherwise} \quad \text{(normal)} \end{cases} \quad (31)$$

Where τ is anomaly threshold. Equation (31) flags every sample as normal or malicious using probability threshold τ, which makes it possible to identify anomalies in real-time.

Algorithm 1: Dynamic Hyper-Parallel Attention Network (DHPAN)

- Input: Network traffic dataset, Learning rates for Adam and RMSprop, Number of training epochs, Anomaly detection threshold
- Initialize CRNN + AAA model parameters
- Initialize Adam optimizer states
- Initialize RMSprop optimizer states
- For each epoch in total_epochs:
- For each mini-batch in dataset:
- Perform preprocessing (normalization, encoding)

- Apply feature selection
- Generate compact embeddings using LFEL
- Extract spatial features using CNN layers
- Apply activation (ReLU) and pooling
- Extract temporal features using GRU layers
- Apply Anomaly-Aware Attention to highlight suspicious patterns
- Compute training loss
- Compute gradients with respect to model parameters
- Update parameters in parallel:
  - Apply Adam optimizer update
  - Apply RMSprop optimizer update
  - Fuse Adam and RMSprop updates
- End for (mini-batch)
- End for (epoch)
- For each sample in test set:
- Generate LFEL embeddings
- Extract features using CRNN + AAA
- Compute class probabilities using fully connected layer and softmax
- Compute anomaly score
- Assign detection label based on threshold:
- If score >= threshold → anomalous
- Else → normal
- End for (test samples)
- Return trained model, anomaly scores, and predicted labels
- Output: Trained DHPAN model, Anomaly scores for test samples, Predicted labels

The algorithm 1 presents the workflow of DHPAN, in which the network traffic is initially preprocessed, features are then selected, and compact embeddings are created through LFEL. The CRNN runs on these embeddings to provide Anomaly-Aware Attention, spatial-temporal patterns, and optimizes model parameters simultaneously with Adam and RMSprop. Lastly, the anomaly scores of all samples of the test are calculated and the detection labels are provided on the basis of threshold of the real-time intrusion detection.

IV. Results and Discussions

The results and discussion section interpolates the experimental findings of the proposed DHPAN framework, and emphasizes its usefulness in real-time intrusion detection on two datasets. It uses baseline models as a reference and the performance and network level metrics to analyze performance and study the effects of major architectural elements by ablation and sensitivity to thresholds. Other practical considerations that it discusses here include latency, throughput, and reliability and this gives a full evaluation of the applicability of DHPAN in the real-life network environment.

A. Performance Evaluation of Intrusion Detection Methods

Performance evaluation helps to evaluate the effectiveness and reliability of the proposed model as it quantitatively measures the accuracy of the proposed model in classifying the network traffic. It allows objective comparison with the available methods working under the same datasets and in experimental conditions. The measures of Performance: Accuracy, Precision, Recall (Detection Rate), F1-score, False Positive Rate (FPR), ROC/AUC and the Confusion Matrix are all popular in networking and intrusion detection systems to fully assess the classification performance. Accuracy gives a general measure of correctness, whereas Precision and Recall give measurements of reliability of attack detection and system to distinguish malicious traffic respectively. The F1-score is Precision-Recall balanced which suits the use of imbalanced network datasets. FPR plays a very important role in the network applications in order to determine the unwanted alarm generation that affects the stability of the networks. ROC/AUC measures the discrimination capacity of the classifier at thresholds and the Confusion Matrix provides a more detailed look at the correct or incorrect packet/flow classification. For this evaluation comparing DHPAN with existing Gradient Recurrent Unit (GRU) Al-Kahtani, M. S., et al. (2023) [26], Convolutional Neural Network (CNN) Cao, B., et al (2022) [27], Recurrent Neural Network (RNN) Mahdi, M. S. (2024) [28], Long-Short Term Memory (LSTM) Induru, V., & Pushpakumar, R. (2020) [29].

	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FPR (%)	AUC/ROC
Dataset 1	GRU	95.21	94.80	94.35	94.57	4.9	0.962
	CNN	96.08	95.72	95.10	95.41	4.2	0.971
	RNN	94.63	94.10	93.78	93.94	5.6	0.954
	LSTM	97.12	96.85	96.40	96.62	3.6	0.982
	DHPAN	99.59	99.41	99.33	99.37	0.48	0.998
Dataset 2	GRU	94.78	94.21	93.89	94.05	5.3	0.958
	CNN	95.86	95.34	94.92	95.13	4.5	0.969
	RNN	94.10	93.65	93.21	93.43	5.9	0.952
	LSTM	96.74	96.30	95.88	96.09	3.8	0.979
	DHPAN	99.68	99.52	99.47	99.49	0.41	0.999

Table 2: Quantitative Metrics Performance Evaluation Comparison Table

Table 2 shows that DHPAN is more successful in processing both datasets and reaches the maximum Accuracy, F1-score, and AUC at the lowest False Positive Rate (FPR). This is a sign of better detection, strength, and consistency of the proposed DHPAN model in various intrusion data.

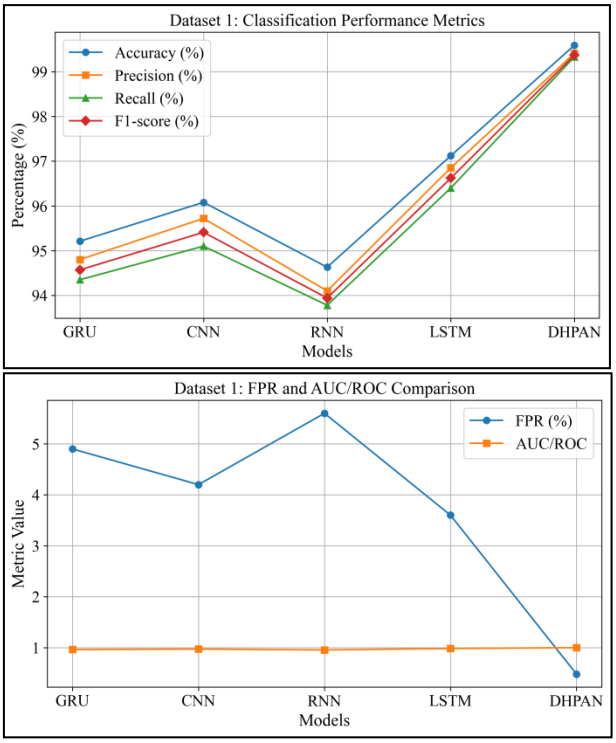

Fig. 5: Performance Evaluation Comparison Chart of Dataset 1

Figure 5 shows the dataset 1 performance evaluation of existing and proposed intrusion detection methods. In this chart the x-axis shows the intrusion detection methods and the y-axis shows the performance metric values.

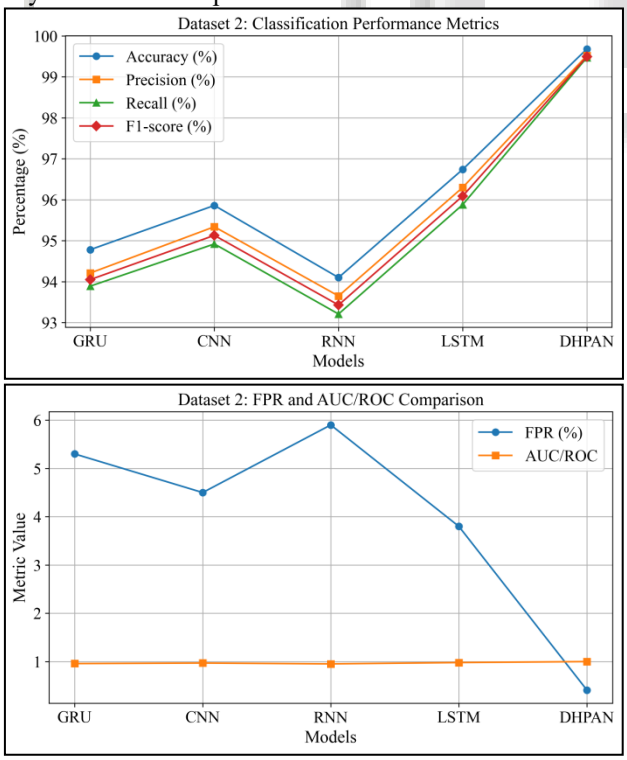

Fig. 6: Performance Evaluation Comparison Chart of Dataset 2

Figure 6 shows the dataset 2 performance evaluation of existing and proposed intrusion detection methods. In this chart the x-axis shows the intrusion detection methods and the y-axis shows the performance metric values.

B. Network-Level Performance and Security QOS Metrics

Besides the measures of classification accuracy, the proposed DHPAN model is measured by the network-level Quality of Service (QoS) measures, such as throughput, detection latency, packet drop rate, detection rate, and packet delivery ratio, to determine its practical implications on real network processes.

		Throughput				
Dataset	Packet Size	GRU	CNN	RNN	LSTM	DHPAN
Dataset 1	20	68.4	71.2	65.9	74.6	82.3
	60	124.7	129.5	118.3	134.9	148.6
	100	176.2	182.8	169.4	188.6	206.9
Dataset 2	20	66.1	69.4	63.8	72.3	80.5
	60	121.9	126.7	116.2	132.8	145.1
	100	172.4	178.6	166.1	185.2	203.7

Table 3: Throughput Comparison Table

In table 3 throughput is measured to get the efficiency of intrusion detection models in different sizes of packets. The proposed DHPAN is always able to achieve better throughput in both datasets owing to optimization of feature learning as well as lowering processing overhead cost proving its applicability in high traffic network settings.

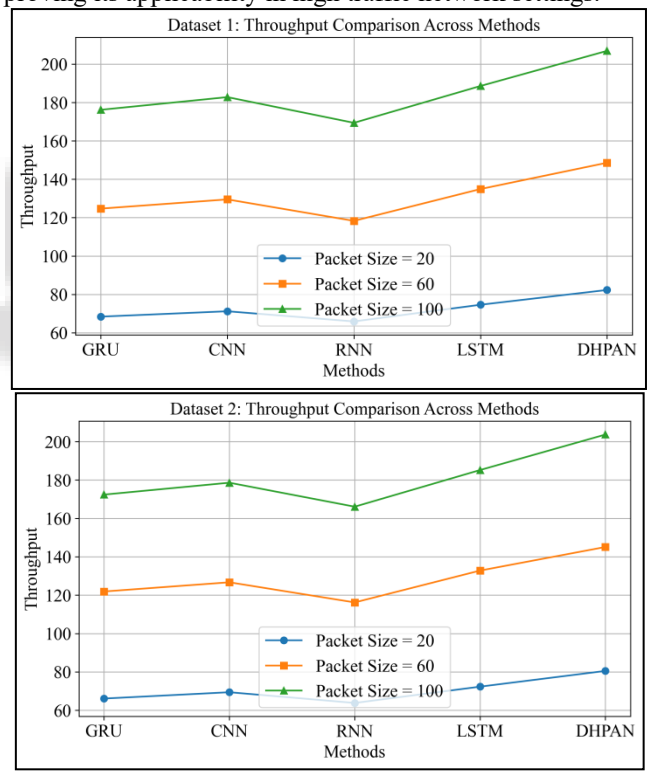

Fig. 7: Throughput Comparison Chart for Dataset 1 And Dataset 2

Figure 7 shows the throughput comparison of intrusion detection methods using dataset 1 and dataset 2. In this chart the x-axis shows the intrusion detection methods and the y-axis shows throughput values across 20, 50, and 100 packet sizes.

		Detection Latency (ms)				
Dataset	Packet Size	GRU	CNN	RNN	LSTM	DHPAN
Dataset 1	20	18.6	21.3	24.9	27.5	14.2
	60	26.4	29.8	33.7	36.9	19.6
	100	34.1	38.6	42.8	46.2	25.3

Dataset 2	20	21.9	24.7	28.5	31.2	16.8
	60	30.6	34.9	39.4	42.8	23.7
	100	39.8	45.1	50.6	55.4	30.9

Table 4: Detection Latency Comparison Table

Table 4 indicates that DHPAN yields the lowest results when using all the packets sizes of both datasets and hence the best performance as compared to the other frameworks (GRU, CNN, RNN, and LSTM). On the rise of the packet size, all methods have higher values, although, DHPAN has a tangible performance edge as it is more scalable and resourceful.

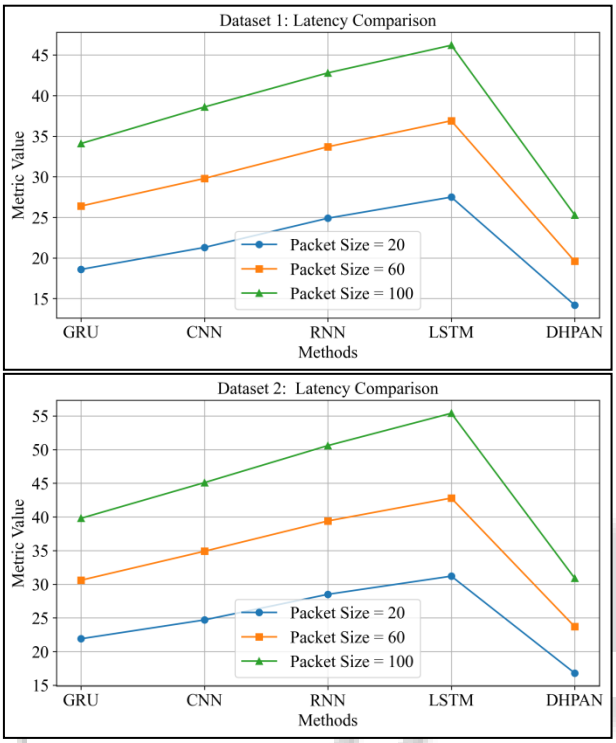

Fig. 8: Latency Comparison Chart

Figure 8 shows the latency comparison of intrusion detection methods using dataset 1 and dataset 2. In this chart the x-axis shows the intrusion detection methods and the y-axis shows latency values across 20, 50, and 100 packet sizes.

	Packet Drop Rate (%)					
Dataset	Packet Size	GRU	CNN	RNN	LSTM	DHPAN
Dataset 1	20	2.9	3.4	4.1	4.6	1.8
	60	4.7	5.3	6.4	7.1	2.9
	100	6.6	7.4	8.9	9.7	4.1
Dataset 2	20	3.6	4.2	5.1	5.8	2.4
	60	5.8	6.6	7.9	8.7	3.6
	100	8.1	9.3	10.8	11.6	5.2

Table 5: Packet Drop Rate Comparison Table

Table 5 illustrates packet size has an upward trend on the packet drop rate of the packet models in both databases and signals network traffic. Nonetheless, DHPAN continues to be the least drop rate in all contexts as it proves to be more reliable and better in congestion management than GRU, CNN, RNN, and LSTM.

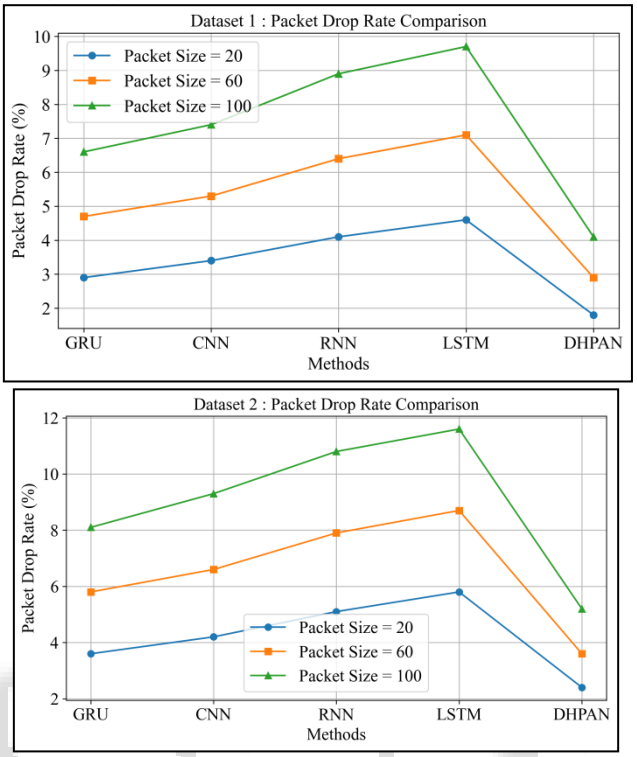

Fig. 9: Packer Drop Rate Comparison Chart

Figure 9 shows the packet drop rate comparison of intrusion detection methods using dataset 1 and dataset 2. In this chart the x-axis shows the intrusion detection methods and the y-axis shows packet drop rate values across 20, 50, and 100 packet sizes.

	Detection Rate (%)					
Dataset	Packet Size	GRU	CNN	RNN	LSTM	DHPAN
Dataset 1 (NSL-KDD)	20	94.6	95.4	93.9	96.8	99.3
	60	93.1	94.2	92.0	95.3	98.9
	100	91.5	92.7	90.4	93.6	98.2
Dataset 2 (CIC-IDS-2017)	20	95.1	96.2	94.0	97.4	99.5
	60	93.7	94.8	92.6	96.1	99.1
	100	92.0	93.3	91.1	94.5	98.6

Table 6: Detection Rate Comparison Table

In table 6 both datasets, the rate of detection reduces with an increase in the size of packets with all baseline models (GRU, CNN, RNN and LSTM) and implies that it is more difficult to classify larger packets. Conversely, the proposed DHPAN model has the highest detection rates (>98) in all packet sizes, which proves good robustness and scalability during different traffic loads.

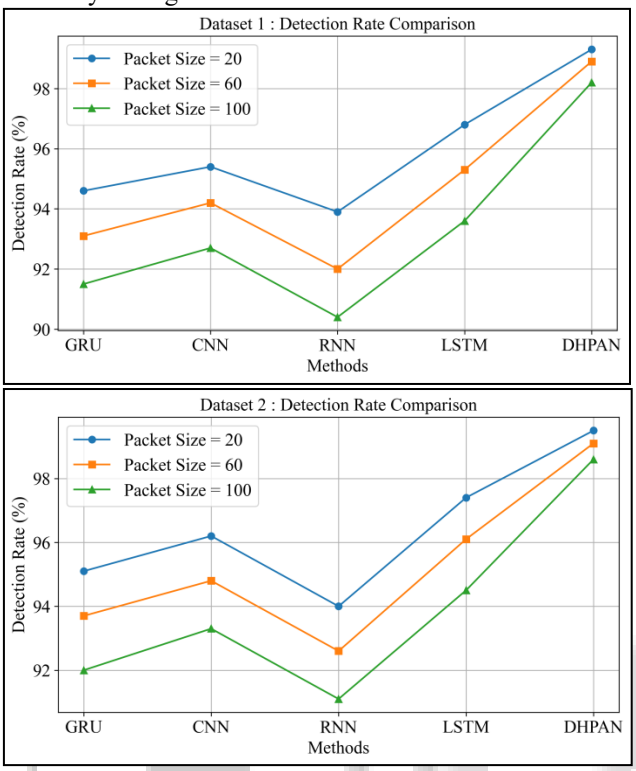

Fig. 10: Detection Rate Comparison Chart

Figure 10 shows the detection rate comparison of intrusion detection methods using dataset 1 and dataset 2. In this chart the x-axis shows the intrusion detection methods and the y-axis shows detection rate values across 20, 50, and 100 packet sizes.

	PDR (%)					
Dataset	Packet Size	GRU	CNN	RNN	LSTM	DHPAN
Dataset 1	20	96.2	97.1	95.4	97.9	99.4
	60	94.8	95.9	93.6	96.5	98.8
	100	93.1	94.3	91.8	95.0	98.1
Dataset 2	20	96.8	97.9	96.1	98.4	99.6
	60	95.3	96.6	94.4	97.2	99.1
	100	93.9	95.0	92.7	95.8	98.5

Table 7: PDR Comparison Table

Table 7 shows the ratio of DHPAN packet delivery remained high when the size of the packet was increased or reduced proves its capability to the network traffic security without affecting the reliability of the communication process.

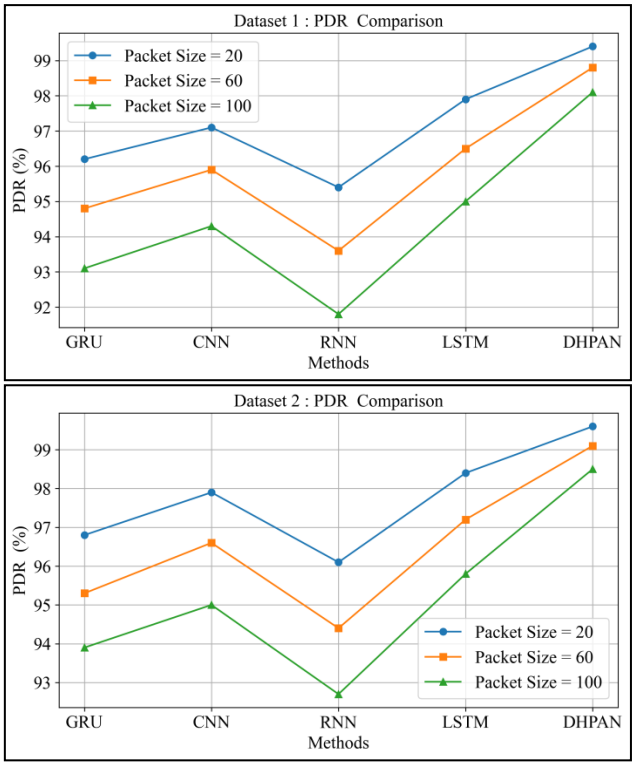

Fig. 11: PDR Comparison Chart

Figure 11 shows the PDR comparison of intrusion detection methods using dataset 1 and dataset 2. In this chart the x-axis shows the intrusion detection methods and the y-axis shows PDR values across 20, 50, and 100 packet sizes.

C. Ablation Study

A study of ablation is done to examine the individual contribution of each key component in the proposed DHPAN framework. Through a progressive elimination or alteration of individual modules, the research points out the contribution of individual elements to the overall performance of the intrusion detection.

Configuration	Dataset 1 Accuracy (%)	Dataset 2 Accuracy (%)
Full DHPAN	99.59	99.68
Without LFEL	97.12	97.34
Without AAA	96.48	96.91
Without CRNN (CNN only)	95.86	96.08
Single Optimizer (Adam only)	98.21	98.39

Table 8: Ablation Study Comparison Table

Table 8 shows that the loss of any of the core components will cause a significant decrease in performance, which proves their relevance. Specifically, LFEL and AAA can be used to improve feature discrimination and anomaly focus at a significant degree, whereas HPDO can be used to improve convergence stability and generalization, which confirms the architectural design of DHPAN.

D. Threshold sensitivity analysis

Threshold Sensitivity Analysis It assesses the sensitivity of the proposed DHPAN model to the selection of the anomaly score threshold (τ). Trade-off between true positive detection and false positive generation (varying τ) would assist us in

selecting the best threshold to use in real-time network intrusion detection.

Threshold (τ)	Dataset 1 Accuracy (%)	Dataset 1 FPR (%)	Dataset 2 Accuracy (%)	Dataset 2 FPR (%)
0.3	98.21	6.5	98.39	6.2
0.4	98.76	4.8	98.91	4.5
0.5	99.12	3.9	99.23	3.6
0.6	99.35	3.1	99.42	2.9
0.7	99.44	2.5	99.53	2.3
0.8	99.40	1.9	99.49	1.8
0.9	99.28	1.2	99.36	1.1

Table 9: Threshold Sensitivity Table

Table 9 shows the reduced thresholds (τ 0.4) are more sensitive with higher false positives; increased thresholds (τ 0.8 or higher) cause fewer false positives but run the risk of failing to detect low profile attacks. The optimal threshold (τ 0.607) is the threshold with high detection rate and low false positive rate that is optimal because it provides efficient real time intrusion detection in both datasets.

V. Conclusion

In this research, the author introduces DHPAN (Dynamic Hyper-Parallel Attention Network) a multimodal deep learning system to detect intrusion in complex cloud and IoT networks in real-time. DHPAN achieves better detecting performance than traditional models by incorporating LFEL to embed compact features, CRNN to extract spatial-temporal patterns with the help of Anomaly-Aware Attention (AAA), and Hyper-Parallel Dual Optimization (HPDO) to train it in a stable way. Empirical evidence of NSL-KDD and CIC-IDS-2017 datasets proves that DHPAN yields the highest accuracy of 99.59% and 99.68%, respectively, and the low rate of false positives. Evaluations at the network level indicate that DHPAN has the best throughput and packet delivery ratio with minimal detection latency and packet drop rates maintaining efficient and reliable real-time processing. Ablation experiments prove that all the parts play important roles in the overall performance and the threshold sensitivity analysis indicates the best anomaly score threshold in the tradeoff between detecting and false alarms. All in all, DHPAN offers a well-developed, scalable, and interpretable intrusion detection solution, which can also detect zero-day attacks, as well as multi-vector attacks, without sacrificing the network QoS. The framework is shown to be practical in the high volume network environment, providing high level of security as well as efficiency in terms of operations. The future research can consider the integration with the edge computing to enable the distributed deployment and adaptive model updating in the live traffic case.

References

[1] Kumar, A., Abhishek, K., Ghalib, M. R., Shankar, A., & Cheng, X. (2022). Intrusion detection and prevention system for an IoT environment. Digital Communications and Networks, 8(4), 540-551.https://doi.org/10.1016/j.dcan.2022.05.027

[2] Attou, H., Mohy-eddine, M., Guezzaz, A., Benkirane, S., Azrour, M., Alabdultif, A., &Almusallam, N. (2023). Towards an intelligent intrusion detection system to detect malicious activities in cloud computing. Applied Sciences, 13(17), 9588. https://doi.org/10.3390/app13179588

[3] Kiran, K. S., Devisetty, R. K., Kalyan, N. P., Mukundini, K., &Karthi, R. (2020). Building a intrusion detection system for IoT environment using machine learning techniques. Procedia Computer Science, 171, 2372-2379.https://doi.org/10.1016/j.procs.2020.04.257

[4] Fatani, A., Dahou, A., AbdElaziz, M., Al-Qaness, M. A., Lu, S., Alfadhli, S. A., &Alresheedi, S. S. (2023). Enhancing intrusion detection systems for IoT and cloud environments using a growth optimizer algorithm and conventional neural networks. Sensors, 23(9), 4430. https://doi.org/10.3390/s23094430

[5] Nizamudeen, S. M. T. (2023). Intelligent intrusion detection framework for multi-clouds–IoT environment using swarm-based deep learning classifier. Journal of Cloud Computing, 12(1), 134.

[6] Lata, S., & Singh, D. (2022). Intrusion detection system in cloud environment: Literature survey & future research directions. International Journal of Information Management Data Insights, 2(2), 100134.https://doi.org/10.1016/j.jjimei.2022.100134

[7] Hazman, C., Guezzaz, A., Benkirane, S., &Azrour, M. (2023). lIDS-SIoEL: intrusion detection framework for IoT-based smart environments security using ensemble learning. Cluster Computing, 26(6), 4069-4083.https://doi.org/10.1007/s10586-022-03810-0

[8] Mohamed, D., & Ismael, O. (2023). Enhancement of an IoT hybrid intrusion detection system based on fog-to-cloud computing. Journal of Cloud Computing, 12(1), 41.https://doi.org/10.1186/s13677-023-00420-y

[9] Onyema, E. M., Dalal, S., Romero, C. A. T., Seth, B., Young, P., &Wajid, M. A. (2022). Design of intrusion detection system based on cyborg intelligence for security of cloud network traffic of smart cities. Journal of Cloud Computing, 11(1), 26. https://doi.org/10.1186/s13677-022-00305-6

[10] Mohamed, R. H., Mosa, F. A., &Sadek, R. A. (2022). Efficient intrusion detection system for IoT environment. International Journal of Advanced Computer Science and Applications, 13(4).

[11] Alsulami, A. A., Abu Al-Haija, Q., Tayeb, A., &Alqahtani, A. (2022). An intrusion detection and classification system for IoT traffic with improved data engineering. Applied Sciences, 12(23), 12336. https://doi.org/10.3390/app122312336

[12] 12. Albulayhi, K., Smadi, A. A., Sheldon, F. T., & Abercrombie, R. K. (2021). IoT intrusion detection taxonomy, reference architecture, and analyses. Sensors, 21(19), 6432. https://doi.org/10.3390/s21196432

[13] De Souza, C. A., Westphall, C. B., Machado, R. B., Sobral, J. B. M., & dos Santos Vieira, G. (2020). Hybrid approach to intrusion detection in fog-based IoT environments. Computer Networks, 180, 107417.https://doi.org/10.1016/j.comnet.2020.107417

[14] Alkhonaini, M. A., Alohali, M. A., Aljebreen, M., Eltahir, M. M., Alanazi, M. H., Yafoz, A., ...&Khadidos, A. O. (2025). Sandpiper optimization with hybrid deep learning model for blockchain-assisted intrusion detection in iot environment. Alexandria Engineering

Journal, 112, 49-62.https://doi.org/10.1016/j.aej.2024.10.032
[15] Hussain, F., Abbas, S. G., Shah, G. A., Pires, I. M., Fayyaz, U. U., Shahzad, F., ...&Zdravevski, E. (2021). A framework for malicious traffic detection in IoT healthcare environment. Sensors, 21(9), 3025. https://doi.org/10.3390/s21093025
[16] Lin, H., Xue, Q., Feng, J., &Bai, D. (2023). Internet of things intrusion detection model and algorithm based on cloud computing and multi-feature extraction extreme learning machine. Digital Communications and Networks, 9(1), 111-124.https://doi.org/10.1016/j.dcan.2022.09.021
[17] Sheelavant, K. K., Yamini, C., Bhushan, P., & Kumar, C. (2025). Ensemble Learning-Based Intrusion Detection and Classification for Securing IoT Networks: An Optimized Strategy for Threat Detection and Prevention. Journal of Intelligent Systems and Internet of Things, 101-118. Doi: https://doi.org/10.54216/JISIoT.170208
[18] Lazzarini, R., Tianfield, H., &Charissis, V. (2023). A stacking ensemble of deep learning models for IoT intrusion detection. Knowledge-Based Systems, 279, 110941.https://doi.org/10.1016/j.knosys.2023.110941
[19] Wang, S., Xu, W., & Liu, Y. (2023). Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things. Computer Networks, 235, 109982.https://doi.org/10.1016/j.comnet.2023.109982
[20] Gaber, T., Awotunde, J. B., Torky, M., Ajagbe, S. A., Hammoudeh, M., & Li, W. (2023). Metaverse-IDS: Deep learning-based intrusion detection system for Metaverse-IoT networks. Internet of Things, 24, 100977.https://doi.org/10.1016/j.iot.2023.100977
[21] Chaganti, R., Suliman, W., Ravi, V., &Dua, A. (2023). Deep learning approach for SDN-enabled intrusion detection system in IoT networks. Information, 14(1), 41. https://doi.org/10.3390/info14010041
[22] Wardana, A. A., Kołaczek, G., & Sukarno, P. (2024). Lightweight, trust-managing, and privacy-preserving collaborative intrusion detection for internet of things. Applied Sciences, 14(10), 4109. https://doi.org/10.3390/app14104109
[23] Roy, S., Li, J., Choi, B. J., &Bai, Y. (2022). A lightweight supervised intrusion detection mechanism for IoT networks. Future Generation Computer Systems, 127, 276-285.https://doi.org/10.1016/j.future.2021.09.027
[24] Alshahrani, H., Khan, A., Rizwan, M., Reshan, M. S. A., Sulaiman, A., &Shaikh, A. (2023). Intrusion detection framework for industrial internet of things using software defined network. Sustainability, 15(11), 9001.
[25] Sharadqh, A. A., Hatamleh, H. A. M., Alnaser, A. A. M. A. A., Saloum, S. S., &Alawneh, T. A. (2023). Hybrid chain: Blockchain enabled framework for bi-level intrusion detection and graph-based mitigation for security provisioning in edge assisted IoT environment. IEEE Access, 11, 27433-27449.DOI: 10.1109/ACCESS.2023.3256277
[26] Al-Kahtani, M. S., Mehmood, Z., Sadad, T., Zada, I., Ali, G., & ElAffendi, M. (2023). Intrusion detection in the Internet of Things using fusion of GRU-LSTM deep learning model. Intelligent Automation & Soft Computing, 37(2), 2279-2290. DOI:10.32604/iasc.2023.037673
[27] Cao, B., Li, C., Song, Y., Qin, Y., & Chen, C. (2022). Network intrusion detection model based on CNN and GRU. Applied Sciences, 12(9), 4184. https://doi.org/10.3390/app12094184
[28] Mahdi, M. S. (2024). Intrusion Detection Systems Based on RNN and GRU Models using CSE-CIC-IDS2018 Dataset in AWS Cloud. Journal of Al-Qadisiyah for Computer Science and Mathematics, 16(4), 141-160. DOI:10.29304/jqcsm.2024.16.41780
[29] Induru, V., & Pushpakumar, R. (2020). Combining LSTM and GRU for efficient intrusion detection and alert correlation in cloud networks. International Journal of Multidisciplinary Research and Growth Evaluation, 1(1), 154-160. https://doi.org/10.54660/.IJMRGE.2020.1.1.154-160

