Vol. 8, No. 3, June 2026

ISI Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

A Hybrid LSTM-GNN-Q-Learning Model for Zero-Day Attack Detection: Evaluation on CICIDS2017 with Simulated Zero-Day Setting

Musa L. Kazimoto[1], Juma S. Ally[2], Stanley Leonard[3]

[1]Department of Information Communication and Technology, Tanzania Institute of Accountancy, Tanzania.

[2]Department of Electronics and Telecommunication Engineering, Mbeya University of Science and Technology, Mbeya, Tanzania.

[3]Department of Information Systems and Technology, Mbeya University of Science and Technology, Tanzania,

Received:
October 15, 2025
Revised:
May 15, 2026
Accepted:
June 6, 2026
Published:
June 27, 2026

Corresponding Author:
Author Name*:
Musa L. Kazimoto
Email*:
musalkazimoto2@gmail.com

DOI:
10.63158/journalisi.v8i3.1670

Abstract. Zero-day attacks exploit previously unseen vulnerabilities, making them difficult to identify using signature-based approaches. Their ability to bypass conventional detection mechanisms can result in significant financial losses, system compromise, and data breaches. To address this challenge, this study proposes a Hybrid Predictive Deep Learning (HPDL) model that integrates the Long Short-Term Memory (LSTM) network for modelling temporal relationships, Graph Neural Networks (GNN) for structural relationship modelling, and Q-Learning for feature weighting and adaptive decision making. The model was evaluated on CICIDS2017 dataset under a simulated zero-day setting by holding out four attack types (Brute Force, SQL Injection, XSS, and Infiltration), totaling 2,179 zero-day samples deliberately excluded from training and validation and used only for final testing. Experimental results show that the proposed HPDL model achieved a zero-day attack detection accuracy of 99.63% and F1-score of 0.9970, outperforming LSTM-only and GNN-only baseline models, which achieved accuracies of 98.5% and 85.0%, respectively. These results indicate that integrating temporal, structural, and reinforcement learning paradigms provides an effective approach for zero-day attack detection.

Keywords: Zero-day attack detection; Hybrid deep learning; Intrusion detection system; Graph Neural Network; Reinforcement learning

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

Zero-day attacks leverage previously unidentified vulnerabilities, making them undetectable to signature-based techniques [1]. Operationally, zero-day detection refers to identifying attacks for which no signature exists, such as new attack classes, new variants or new exploitation methods [2], [3]. With this "day zero" knowledge gap, attackers can infiltrate the system, steal valuable information, and send malware, resulting in considerable financial and reputational damage [4].

Recent attacks, including zero-day vulnerability discovered in popular WordPress plugins [5], have demonstrated that zero-day attacks are serious threats. Studies indicate that zero-day attacks occur every 17 days on average, remain undetected for approximately 312 days, and account for 80% of security breaches, with an average cost of $1.2 million per attack [6]. The increasing availability of zero-day exploits purchased from underground markets presents new challenges to the already increasing challenges of zero-day attacks [7]. The challenge posed by zero-day attacks is for the attacker to find and exploit only one zero-day vulnerability, while the defenders must secure themselves against all potential entry points for attackers [8].

The traditional methods of antivirus and anti-malware software, which utilize signature-based and heuristic-based models to detect incoming threats, are often unable to recognize new forms of attack, often creating false positives that can mask the important alerts regarding zero-day attacks [9],[10]. More sophisticated approaches utilizing Deep Learning and Machine Learning can learn for themselves of the various attack vectors that exist within a network. The problem with these systems is that they are static models that cannot adapt to new forms of attack - their performance tends to decline after any alterations to the network or systems within that network, referred to as model drift [11], [12]. The best static models have an accuracy of less than 95% and a false-positive rate that is more than 10% in real-world zero-day environments [13], [14]. Dynamic Machine Learning (DML) and hybrid models are new solutions arising from the demand for more resilient systems. However, these approaches still face limitations, including high false alarm rates and limited generalization across different attack types [15]. In particular, LSTM models are effective at capturing temporal dependencies, while

neglecting the structural dependency between network entities [16]. GNN-based models effectively capture topological relationships, but do not perform temporal sequence analysis [17]. Reinforcement learning methods are adaptive but generally used for static feature representations without temporal and/or structural fusion [18], [19]. Most existing solutions focus on modelling temporal patterns through LSTM or structural relationships through GNN, while comparatively few incorporate decision-making mechanisms [20]. In addition, the reviewed literature indicates limited attempts to integrate the temporal analysis, structural relationship modelling, and adaptive decision-making in a unified framework for zero-day attack detection.

In order to overcome the limitations described above, we propose a Hybrid Predictive Deep Learning (HPDL) model that combines the LSTM model with the Graph Neural Networks (GNN) and Q-Learning models. The main goal of this research is to propose a deep learning model that combines the LSTM, GNN and Q-Learning models for the detection of zero-day attacks in a simulated zero-day scenario using the CICIDS2017 dataset. In particular, four attack classes (Brute Force, SQL Injection, XSS, and Infiltration) were deliberately excluded from training and validation to create realistic zero-day conditions. This hold-out configuration requires the model to detect these attacks without prior exposure to them during training and validation, reflecting a real-world scenario in which no signatures or training examples are available. This study makes the following primary contributions:

1) A unified parallel architecture integrating LSTM (temporal), GNN (structural), and Q-Learning (reinforcement learning) for zero-day attack detection.
2) A realistic zero-day simulation protocol on the CICIDS2017 dataset in which four attack classes (Brute Force, SQL Injection, XSS, and Infiltration) are withheld from training and validation.
3) A comparative evaluation against unimodal baselines (LSTM-only and GNN-only), demonstrating that the proposed HPDL model achieved 99.63% zero-day detection accuracy on CICIDS2017.

The following research questions are formulated to address the identified research gap:

1) RQ1: How effectively can the proposed HPDL model detect zero-day attacks?

2) RQ2: Does Q-Learning improve detection performance through dynamic weighting compared to static fusion of LSTM and GNN?
3) RQ3: How does the proposed HPDL model perform under a simulated zero-day setting compared with unimodal baselines?

2. RELATED WORKS

This section details the foundational models underpinning this research and critically reviews prior work in detection of zero-day attack to clearly establish the contribution of our proposed HPDL model.

2.1. Foundational Machine Learning Models

2.1.1. Long Short-Term Memory Networks

A special class of recurrent neural networks (RNN) called Long Short-Term Memory (LSTM) networks are used to model temporal sequences and to capture long-range dependencies [21]. This capability is very important when dealing with network traffic, as attacks will typically be a series of events that are happening over time. The key component to an LSTM cell is its gating mechanism, which controls the flow of information through the cell. The most important calculations at timestep t are as shown in Equation 1 to 6.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad \text{(Forget Gate)} \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)} \tag{2}$$

$$\widetilde{C} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)} \tag{5}$$

$$h_t = o_t \odot tanh(C_t) \quad \text{(Hidden State)} \tag{6}$$

Where f_t is the forget gate activation vector (with values between 0 and 1). It determines the proportion of the previous cell state C_{t-1} to be retained. i_t is the activation vector of the input gate (with values between 0 and 1). It determines the amount of the candidate cell state $\widetilde{C}$ to be added to the cell state. o_t is the activation vector of the output gate (with values between 0 and 1). The sigmoid activation function σ outputs values between 0 and 1. The symbol $\odot$ denotes the element-wise multiplication of the

two vectors. The input to the LSTM model is denoted as x_t and the hidden state is denoted as ht. The bias vectors b_f, b_i, b_o, b_c, are associated with the forget, input, output and candidate cell state gates, respectively. The weight matrices W_f, W_i, W_o, and W_c are associated with the forget, input, output and candidate cell state gates, respectively. The vector $[h_{t-1}, x_t]$ is formed by concatenating the previous hidden state h_{t-1} and the current input x_t. Finally, the value C_t represents the state of the cell [22]. The LSTM model architecture allows the model to learn from the sequential nature of the network data and recognize the specific patterns associated with the existence of multi-stage network intrusions.

2.1.2. Graph Neural Networks

The GNN model transforms network traffic into a graph whose nodes represent network endpoints and whose edges represent network traffic flows with attributes indicating traffic statistics [23]. Each node in the graph updates its representation through a process that aggregates messages from its neighbours, which allows the node to account for both traffic statistics and the overall traffic structure within the network. A compact formulation of one round of message-passing and update for node v is as shown in Equation 7.

$$h_v^k = \sigma\left(W^{(k)} \left(h_v^{(k-1)} + \Sigma_{u \in N(v)} \emptyset\left(h_u^{k-1}, e_{u,v} \right) \right) + b^k \right) \quad (7)$$

Where is h_v^k the node embedding at iteration k, $N(v)$ denotes the neighbors of v, $e_{u,v}$ are edge features encoding flow statistics, $\emptyset(\cdot)$ is a learnable message function (e.g., an MLP that fuses neighbor embeddings with edge features), $W^{(k)}$, $b^{(k)}$ are layer parameters, and σ is a nonlinearity. After K iterations a readout function (e.g., global pooling or a node-wise classifier) maps the learned embeddings to detection scores for benign vs. malicious traffic [24]. This enables GNNs to capture the structural dependencies in network traffic that are often invisible to traditional models [25].

2.1.3. Q-Learning and Dynamic Weighting

The Deep Q-Network (DQN) algorithm uses Deep Q-Network (DQN), an extension on convectional Q-Learning that uses a neural network to predict the action values, to learn the values of the actions taken in a given state [26], [27]. The aim is to learn the best action-value function for every state and action. The action-value function for an action

to be taken in a state shows the expected cumulative discount reward for taking that action and then following the optimal policy from that state onwards. The standard Q-Learning update rule which underpins DQN is as shown in Equation 8.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\, r_t + \gamma \max_{a'} Q\,(s_{t+1}, a') - Q(s_t, a_t)] \tag{8}$$

where α is the learning rate and γ is the discount factor. This approach enables adaptation to new threat patterns without requiring full model retraining, a critical capability for countering evolving zero-day attacks [28]. The "state" s_t is defined as the fused feature representation generated from the LSTM and GNN branches, while the "action" a_t corresponds to assigning a dynamic weight vector to these features.

This formulation is able to dynamically highlight the features that are more discriminative, depending on the specific context of the input, from either the temporal (LSTM) or the structural (GNN) branch. Learning a policy for feature weighting instead of using fixed connections provides a form of constant light-weight adaptation in learning, making the model more robust to new threats [29].

2.2. Critical Analysis of Related Works in Zero-Day Detection

The field of zero-day attack detection has progressed significantly since the first time the concept was introduced, whereby each new development has addressed some of the issues it faced, while introducing new ones. From a historical viewpoint on those studies that have been appraised against the CICIDS2017 benchmark, these progressions and remaining gaps can be seen. One of the most influential methods proposed was [30] which used supervised learning for detecting zero-day malware using API call signatures. This study demonstrated the possibility to detect unknown threats via machine learning. Early ML models proved to have high false alarms and poor generalization, however, due to its dependence on static analysis and simple classifiers.

Subsequently, the study started to include more complex data types. [31] introduced an approach to discover complex relationships in network traffic based on graph-based features and autoencoders. This was an important milestone in the process of detection with structural data. But the method was complicated, and crucially, could not determine

the sequence and timing of attacks, called multi-stage intrusions, which are vital in the detection of these attacks. The following year, [32] presented a Deep Q-Network-based active learning framework for detecting zero-day attacks. However, their method still relies on separate BiLSTM components for rather than integrating temporal analysis temporal reasoning directly into the Q-Learning decision process. This separation means the Q-learning agent alone does not learn temporal dependencies.

In 2024, [33] have been studying and extending the limits to the reasoning-based framework for IoT devices that work based on deep learning and symbolic logic. It is useful for detecting novel threats, but requires lots of labeled data and does not have a continuous learning mechanism. At the same time, [34] created a federated learning system that achieves 96% accuracy on a balanced dataset but dropped to 59.50 when tested on CICIDS2017 imbalanced dataset and does not employ structural (GNN) or advanced temporal (LSTM) analysis. In the evaluation of their model on the CICIDS2017 dataset, they found that their model was able to capture the spatial and temporal patterns, but were missing network entity relationships and a dynamic weighting mechanism [35].

What is clear in the literature is that a model has been developed that is effective for parts of the problem, some of which are able to cope with time, others with structure, and others with adaptability. Based on the literature reviewed, only a few approaches have been evaluated on CICIDS2017 and have been able to integrate all three of these fundamental features into a single, comprehensive system. The proposed HPDL model aims to address this gap.

3. METHODS

This section explains how network traffic is analyzed using the proposed HPDL model, as illustrated in the conceptual framework in Figure 1. The process begins with the collection of raw datasets which are then cleaned and prepared for both the temporal and structural models. The prepared datasets are fed into the models to train and validate the models to detect and mitigate zero-day attacks. The learning and feedback mechanisms implemented into the system enhance the system's learning aspect to

accommodate for zero-day attacks and eventually creates a more resilient cybersecurity system for the networks in question.

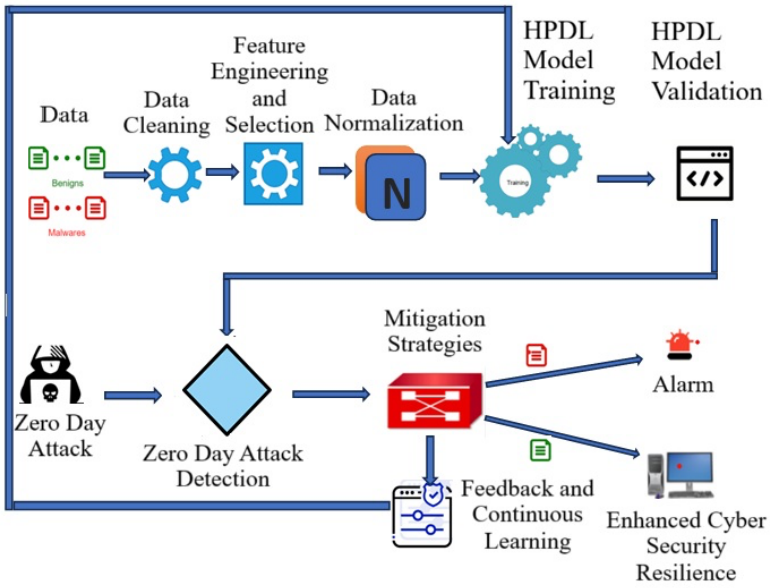

Figure 1: Conceptual Framework of a DML for mitigating Zero-day attacks

3.1. Data Collection and Description

The Canadian Institute of Cybersecurity Intrusion Detection System 2017 (CICIDS2017) dataset, a reference dataset with a simulated and realistic environment of contemporary network, is used in this study. The CICFlowMeter was used to convert five days of network traffic data, which included both normal traffic and various contemporary attacks, into an organized network flow format [36]. Eight full CSV files were programmatically downloaded out of the official repository using a configured *wget* command on a Unix based system and give complete coverage of the work week. The total number of samples used is 2.8 million, 78 flow based features and 1 label are the 15 classes (14 attacks + Benign). This can be seen in Figure 2, which is a breakdown of the attacks, sorted by the frequency of the attack in each category with 557,646 instances of attacks. The significantly imbalanced distribution of these attack types is representative of the real world. Figure 3 displays the distribution of each attack category within the data set. As is visible in the graph, 80% of the data is comprised of non-attacking traffic, while the attacks represent the remaining 20% of the dataset - a 4.08:1 ratio of benign to malicious traffic. For these statistics and additional information regarding each attack category, refer to Table 1.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

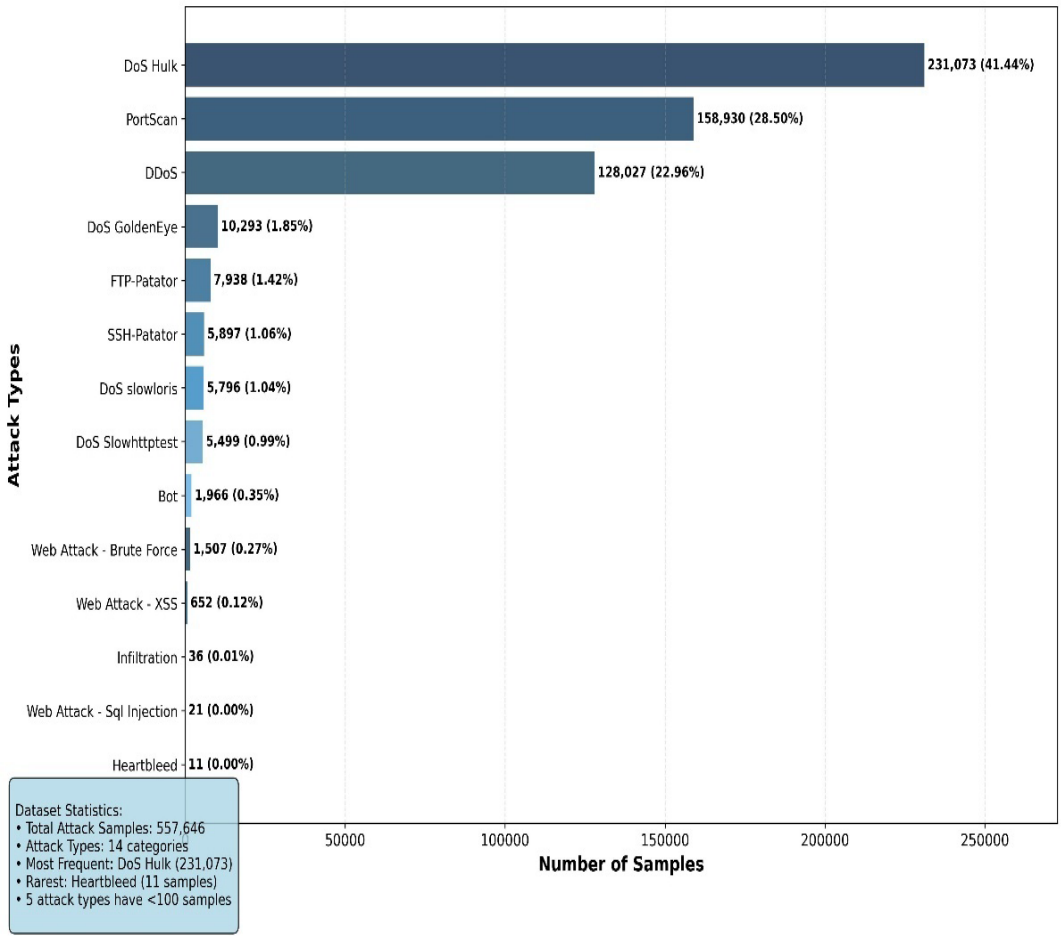

Figure 2. Attack type frequency distribution (14 categories, 557K attack samples)

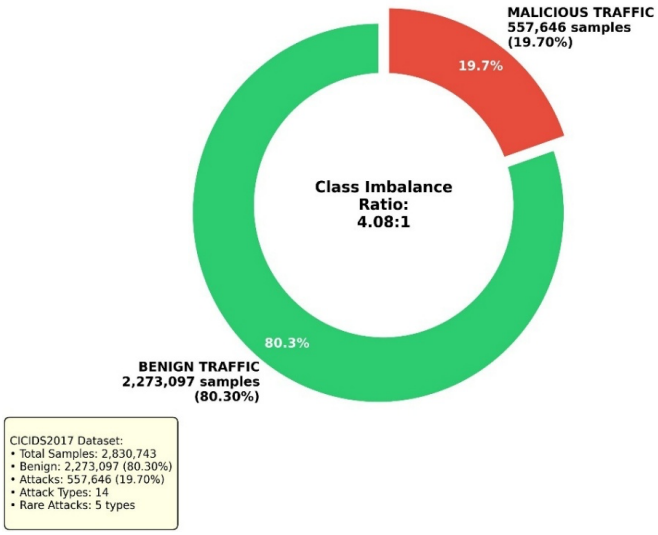

Figure 3. Class imbalance: 80% benign vs 20% attacks (4.08:1 ratio)

The characteristics of the CICIDS2017 dataset, including its realistic network simulation, variety of attack types, and class imbalance, make it well-suited to evaluating the proposed HPDL model.

Table 1: Raw CIC-IDS2017 Dataset Characteristics (79 Features)

File	Rows	Missing Values	Infinite Values	Duplicates
Monday-WorkingHours	529,918	1	27	86
Tuesday-WorkingHours	445,909	9	17	35
Wednesday-workingHours	692,703	0	14	34
Thursday-WorkingHours-Morning-WebAttacks	170,366	0	10	45
Thursday-WorkingHours-Afternoon-Infilteration	288,602	0	0	54
Friday-WorkingHours-Morning	191,033	0	10	82
Friday-WorkingHours-Afternoon-DDos	225,745	0	8	26
Friday-WorkingHours-Afternoon-PortScan	286,467	0	2	67
TOTAL	**2,830,73**	**10**	**88**	**429**

3.2. Data Preparation

Strong data cleaning process was put in place to assure data quality and integrity in the datasets for analysis. Multiple steps were needed in order to solve the typical problems with network traffic data. All string fields were trimmed of extra whitespace and non-printable characters, in order to avoid parsing problems. Features that had more than 70% null values were dropped because they were not reliable features, and the null and infinite values were replaced with the median values of the features, a robust measure which would not be affected by outliers from the data distributions that were expected in the network. In addition, all rows were removed if they were identical to avoid patterns in the model. Lastly, the index of the DataFrame was reset after the elimination of the rows to normalize and continue with the index. This comprehensive and integrated cleaning effort resulted in a streamlined, high-integrity cleaned dataset of the

characteristics described in the following table: Cleaned CIC-IDS2017 Dataset Characteristics. A stripped-down version using business terminology and clearly leading the reader to the table to obtain the final outcome of the cleaning process.

3.2.1. Dataset Cleaning

Data has been cleaned through a robust data cleaning pipeline to guarantee the quality and integrity of the data. It has several steps to deal with frequent issues with network traffic data. Extraneous white space characters (including non-printable) were removed from all string fields to avoid parsing problems. Feature reliability was maintained by discarding all the null values in more than 70% of the columns, with the rest of the values filled in with the median of each respective feature, which is a robust measure that is less affected by outliers common in network data distribution. Furthermore, the bias of the model was eliminated by removing all duplicate rows. Lastly, the index of the DataFrame was reset so as to give a consistent and continuous indexing following row deletion. The comprehensive removing process gave rise to a refined and high integrity dataset as summarized in Table 2.

Table 2: Cleaned CIC-IDS2017 Dataset Characteristics with 79 Features

File	Rows	Missing Values	Infinite Values	Duplicates
Monday-WorkingHours	502,983	0	0	0
Tuesday-WorkingHours	421,844	0	0	0
Wednesday-WorkingHours	610,794	0	0	0
Thursday-WorkingHours-Morning-WebAttacks	164,300	0	0	0
Thursday-WorkingHours-Afternoon-Infiltration	252,972	0	0	0
Friday-WorkingHours-Morning	184,145	0	0	0
Friday-WorkingHours-Afternoon-DDoS	223,112	0	0	0
Friday-WorkingHours-Afternoon-PortScan	214,114	0	0	0
TOTAL	**2,574,264**	**0**	**0**	**0**

3.2.2. Feature Engineering and Selection

To give the model increased discriminative power and computational efficiency, a memory-efficient feature engineering and selection pipeline was implemented. The feature engineering process created three new features using domain knowledge: (1) flow packet ratio, (2) average packet size, and (3) protocol-service combination. Only the training set was subjected to a two-stage method for feature selection. The highest-ranking features were chosen as candidates after all features were ranked using Information Gain Ratio (IGR) filtering. Second, the least significant features were eliminated using the Random Forest classifier and the RFECV algorithm. The ideal feature set, $F_{optimal}$, was determined by selecting the features with the highest cross-validation score.

3.2.3. Data Transformation and Encoding

The dataset was transformed and encoded to ensure numerical stability in gradient-based learning and to handle categorical data appropriately. To reduce the impact of extreme values, outlier capping was implemented at the 1st and 99th percentile using only the training set. The StandardScaler was applied to normalize each feature to have a zero mean and unit variance. The scaler was fitted on the training set only and then used for transforming the validation and test set. The LabelEncoder was fitted on the training set to encode the categorical attack labels. The complete mapping of the original attack labels to the final categories is presented in section 3.2.4 (Table 3).

3.2.4. Zero-Day Simulation and Data Split

The study designed a data-split technique to realistically test the model's ability to detect previously unseen attacks called, referred to this study as simulated zero-day attacks. To do this, the study completely held out four of attack types — WebAttacks (Brute Force, SQL Injection, and XSS), and Infiltration from the start as illustrated in Figure 4. Before any label grouping or dataset splitting, the hold-out procedure was performed immediately after the data cleaning stage. Four attack classes (Web Attack-Brute Force, Web Attack-SQL Injection, Web Attack-XSS, and Infiltration) were excluded from training and validation and reserved for final testing to simulate zero-day conditions. The final

classes used in this study were then created by grouping the remaining attack classes. The mapping between the original CICIDS2017 labels, final categories, and their corresponding dataset usage is presented in Table 3.

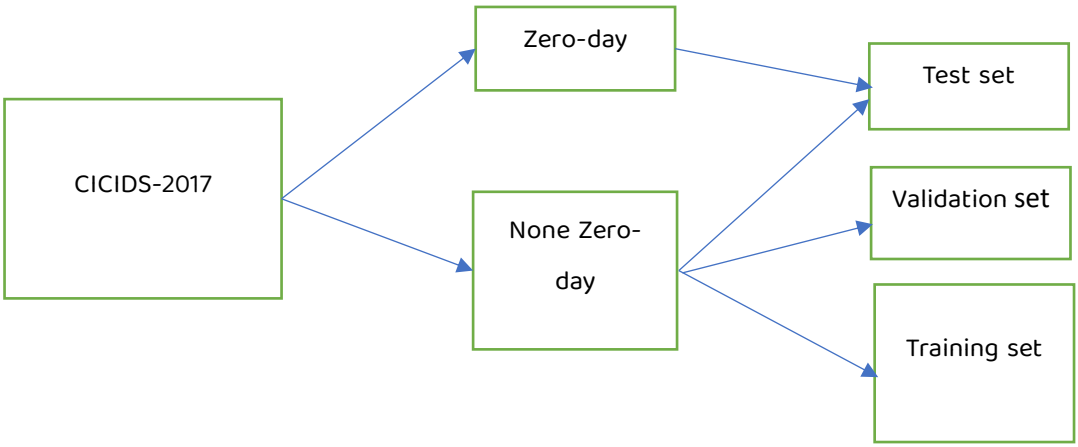

Figure 4: Dataset split and simulation of zero-day attack traffic

Table 3: Mapping of CICIDS2017 labels to final classification categories

Original CICIDS2017 Label	Final Category	Dataset Usage
BENIGN	Normal	Train / Validation / Test
FTP-Patator	FTP, SSH, Heartbleed	Train / Validation / Test
SSH-Patator	FTP, SSH, Heartbleed	Train / Validation / Test
Heartbleed	FTP, SSH, Heartbleed	Train / Validation / Test
DDoS	DDoS, PortScan	Train / Validation / Test
PortScan	DDoS, PortScan	Train / Validation / Test
Bot	Bot	Train / Validation / Test
DoS Hulk	DoS	Train / Validation / Test
DoS GoldenEye	DoS	Train / Validation / Test
DoS Slowloris	DoS	Train / Validation / Test
DoS Slowhttptest	DoS	Train / Validation / Test
Web Attack – Brute Force	Zero-Day Attack	Test Only
Web Attack – SQL Injection	Zero-Day Attack	Test Only
Web Attack – XSS	Zero-Day Attack	Test Only
Infiltration	Zero-Day Attack	Test Only

Table 4 presents the sample counts for each hold-out zero-day class before and after the data cleaning processes described in section 3.2.1. After cleaning (removal of duplicates, missing values, and infinity values), the total zero-day samples decreased from 2,216 to 2,179.

Table 4: Sample counts for excluded zero-day classes before and after data cleaning

Zero-Day Class	Before Cleaning	After Cleaning
Brute Force	1,507	1,470
SQL Injection	21	21
XSS	652	652
Infiltration	36	36
TOTAL	**2,216**	**2,179**

The model never saw zero-day traffic during training and validation. The rest of the data (known attacks and normal traffic) was divided into three parts: 70% for training, 15% for validation, and 15% for testing as detailed in Table 5. Finally, the study added the held-out zero-day attacks into the test set. This method creates a challenging and realistic test. The model must find the known attacks as well as discover new attacks on the system during the final exam to exhibit its usefulness in the wild.

Table 5: Final data set distribution following the strategic 70/14.9/15.1 split for zero-day simulation

Split Type	Samples	Percentage	Purpose
Training Set	1,800,456	70%	Model Training
Validation Set	385,813	14.9%	Hyperparameter Tuning
Testing Set	387,986	15.1%	Final Evaluation

Recent studies have viewed zero-day detection as an open set recognition task that requires the model to not only recognize known classes, but also reject unknown classes [37]. This view is consistent with our 'hold-out' simulation approach.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

3.2.5. Addressing Class Imbalance

The given dataset was imbalanced between the benign and attack traffic. This could lead to biases in the trained model. To reduce the effect of this, SMOTE was applied only to the training set. SMOTE was applied after the transformation and scaling steps but before the construction of sequences and graphs. SMOTE helps to create synthetic examples of the minority class by interpolating between the existing instances of that class. This ensures that the distribution of training samples is made more uniform as shown in table 6, while maintaining the original and realistic distributions of the samples in the validation and test set.

Table 6: Training set before and after applying SMOTE

Class	Before	After	Change	Percentage
Normal	1,503,870	1,503,870	+0	71.30%
Bot	1,367	13,670	+12,303	0.60%
FTP, SSH, Heartbleed	6,406	51,248	+44,842	2.40%
DDoS, PortScan	89,611	268,833	+179,222	12.80%
DoS	135,624	271,248	+135,624	12.90%
TOTAL	1,736,878	2,108,869	371,991	100%

3.2.6. Sequences Preparation for HPDL Modelling

Network Traffic data were transformed into multi-modal inputs for HPDL, creating temporal sequences for LSTM analysis and graph structure for GNN processing while implementing a reward-based Q-Learning system optimized for zero-day detection. The 10-time step sequences were constructed using sliding window (stride = 1) over consecutive network flow, where each time step contains 52 features from $F_{optimal}$ per flow. A window size of 10 captures temporal context of multistep attacks generating approximately 1,790,972 training sequences, 385,804 validation sequences and 387,986 test sequences as shown in the Table 7. Thus, overlapping windows result in a reduced sequence count.

The graph was constructed using cosine similarity of node features. In this study, graph nodes represent sampled feature vectors from the dataset. In order to capture the

primary connectivity patterns with reasonable memory and training time, a graph size of 200 nodes was selected to balance structural representation and computational efficiency. Edges were created between node pairs with a similarity threshold greater than 0.1, and edge weights were scaled to a range of 1.0 to 9.0 based on similarity values. A static graph was built from the entire dataset for each split (training, validation, test) to ensure consistent embeddings across all data partitions. To avoid any data leakage, node selection was based only on feature similarity without the use of label information. This resulted in 200 nodes with 987, 983, and 981 edges, respectively, as summarized in Table 7.

Table 7: Sequence prepared for HPDL Architecture training, validation and testing

Component	Training	Validation	Test	Specifications
Sequences	1,790,972	385,804	387,986	Shape: (10, 52)
Graph Nodes	200	200	200	Features: 52
Graph Edges	987	983	981	Weight range: 1.0-9.0
Normal (1.0)	1,503,862	322,249	322,249	BENIGN traffic
Known Attack (1.5)	287,110	63,555	63,558	Training-known attacks
Zero-Day Attack (3.0)	0	0	2,179	Unseen attacks

3.3. The Proposed HPDL Architecture Model Design

The proposed HPDL model as shown in Figure 5 combines temporal, structural, and reinforcement learning mechanisms into an integrated adaptive detection system for zero-day attacks. Each of the architectures is a distinct analytical task that feeds into a policy network known as Q-Learning, which takes the input from all the tasks to make a decision. The process starts with data preprocessing which is concerned with structuring pre-processing of input traffic data and producing structured inputs which are appropriate for sequential and graph-based learning, as discussed in Section 2.3. For the temporal component, a two-layer LSTM network is used to model the relationships between the traffic windows in time.

The input to the LSTM is a matrix of a sequence of data, $X_{seq} \in \mathbb{R}^{10\times52}$, representing ten consecutive time steps with fifty-two statistical and behavioral traffic features per step. The LSTM layers are configured with 64 and 32 hidden units, respectively, each followed

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

by dropout regularization to mitigate overfitting. The computations within the LSTM cell follow Equations (1)– (6), where the hidden state h_t and cell state C_t jointly preserve long-term behavioral context. The output from the final LSTM layer is a 16-dimensional temporal embedding vector h_t^{LSTM}, representing learned temporal dependencies that reflect evolving network behaviors and potential intrusion sequences.

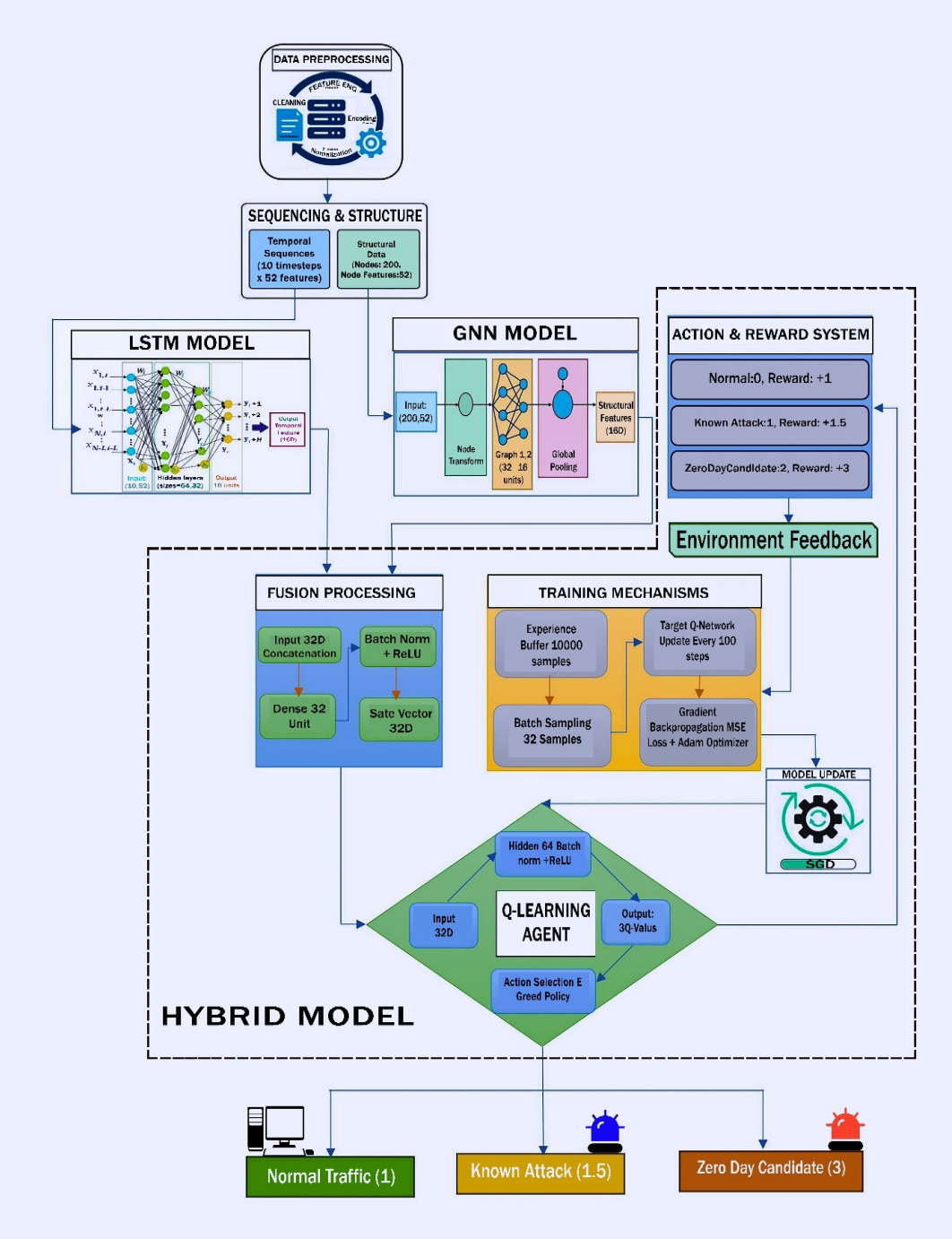

Figure 5: The Proposed Hybrid Predictive Deep Learning (HPDL) Model

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Simultaneously, the structural component leverages a two-layer GNN to model communication relationships among network entities. The input to the GNN is represented as a graph $G = (V, E)$, where each node $v_i \in V$ encodes 16 structural features such as degree, connection frequency, and packet ratio statistics, while edges $e_{(u,v)} \in E$ capture relational flow features between communicating endpoints. Each GNN layer applies message-passing and aggregation as formulated in Equation (7), with 32 and 16 hidden units, respectively. A global mean pooling layer aggregates node-level embeddings into a single 16-dimensional graph-level vector h_t^{GNN}, which captures both local connectivity patterns and overall structural dependencies in network traffic. These outputs create the important inputs for the fusion processes.

The temporal and structural embeddings are then passed to the fusion process, which integrates them into a unified latent space. The fused state representation is computed as shown in Equation 9.

$$s_t = \Phi_f([h_t^{LSTM}; h_t^{GNN}]) \tag{9}$$

where Φ_f represents the fusion transformation network, and $([h_t^{LSTM}; h_t^{GNN}]$ denotes the concatenation of temporal and structural embeddings. By integrating the topological and temporal features of network traffic into a shared latent space, this fusion process gives the reinforcement learning agent comprehensive state description. The fusion transformation network employs a fully connected layer, batch normalization and ReLU activation function after it, as shown in Equation 10.

$$f_t = \text{ReLU}\left(\text{BatchNorm}\left(W_f[h_t^{LSTM}; h_t^{GNN}] + b_f\right)\right) \tag{10}$$

where W_f and b_f are learnable parameters and bias vector of the fusion layer respectively. The resulting $f_t \in \mathbb{R}^{32}$ captures both temporal and structural information present in the network traffic data. Since f_t represent the output of the fusion transformation $\Phi_f(.)$, the reinforcement learning state is defined as shown in Equation 11.

$$s_t = f_t \tag{11}$$

The Deep Q-Network (DQN) agent is then given the resultant state vector as an input for policy optimization and action selection.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

The Q-learning agent operates on the fused state space to learn an optimal decision policy. The action-value function for a given state and action represents the expected cumulative reward for performing action in state until the end of the episode. The update rule for the Q-learning mechanism, shown in Equation (8), enables the agent to iteratively update the action-value function until maximizes the cumulative reward earned by the agent over time.

Then, the action and reward system evaluates the agent's classification decisions and assigns feedback to guide further learning. The reward is computed as shown in Equation 12.

$$r_t = \begin{cases} +3, & \text{if correctly identifies a zero} - \text{day attack} \\ +1.5, & \text{if correctly identifies a known attack} \\ +1, & \text{if correctly classifies normal traffic} \\ -2, & \text{for false negatives (malicious traffic incorrectly classified as normal)} \\ -1, & \text{for false positives (normal traffic incorrectly classified as malicious)} \end{cases} \quad (12)$$

This feedback will allow the model to prioritize high-value network traffic (e.g., zero-day attacks) while rejecting low-value network traffic. Additionally, the model can learn from its environment through this feedback loop. When new network traffic is encountered, each branch of the model will receive the data D_{new} and produce s_{t+1}. The policy network will then be updated according to Equation 13.

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta) \quad (13)$$

During training mechanism, the agent stores the experience tuples (s_t, a_t, r_t, s_{t+1}) in a buffer and samples batches from the buffer to calculate the loss function periodically. The loss function $\mathcal{L}(\theta)$ for Deep Q-Network (DQN) is as shown in Equation 14.

$$\mathcal{L}(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})}[\,(r_t + \gamma \max_{a'} Q\,(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta))^2] \quad (14)$$

where, θ and θ^- represent the parameters of the online and target networks respectively. Stochastic gradient descent (SGD) is used to update the model parameters to minimize $\mathcal{L}(\theta)$, thus making sure that convergence is stable.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

The trained Q-network provides the action a_t in $a_t \in \{Normal, Known\ Attack, Zero-Day\ Candidate\}$, that is the classification decision of the model. The policy chosen maximizes the expected reward in the long-run provided by the mathematical as shown in Equation 15.

$$\pi^*(s_t) = arg\max_a Q(s_t, a) \tag{15}$$

where $\pi^*(s_t)$ is the optimal policy that chooses the most desirable action given the state s_t by maximizing the learned Q-value $Q(s_t, a)$. This makes sure that the agent always makes the most rewarding classification decision, depending on the experience learnt.

The learning process is designed as an episode-based training cycle ($episode = 1, \ldots., E_{max}$), in which training is stopped when the convergence requirements are met and model performance is continuously evaluated using validation process. To further stabilize the learning process, the online network parameter θ and the target network parameter θ^- are regularly synchronized. Through this training process, the Q-Learning component continuously updates its action-selection policy in response to the observed rewards, thereby learning an adaptive classification policy. After training, the learned policy is fixed and used for inference without additional parameter adjustment throughout testing and deployment. Code 1 provides a summary of these operations' execution workflow.

```
Input:
D = Network dataset
Θ = Model parameters
C = Training configuration
Output:
M = Trained HPDL model
P = Performance metrics
S_adapt = Adaptation status
function HPDL (D, Θ, C)
Validate and prepare input data
if data is not in hybrid format then
    Generate temporal sequences
    Construct network graph
    Encode labels
end if
for episode = 1 to E_max do
    Z_temp ← LSTM(X_seq)
    Z_struct ← GNN(G_graph)
    S ← Fusion(Z_temp, Z_struct)
    Select action using ε-greedy policy
```

```
    Compute reward
    Store experience in replay buffer
    Sample experiences from replay buffer
    Update Q-network using experience replay
    if target update condition is satisfied then
        Synchronize target network parameters
    end if
    Validate model performance
    if convergence criterion is met then
        break
    end if
end for
while deployment is active do
    Collect new network traffic
    Update graph structure
    Extract hybrid features
    Classify network state using trained policy
end while
return M, P, S_adapt
end function
```

Code 1. HPDL Training and Adaptive Inference

3.4. Modal Evaluation

This part will compare three models: the baseline LSTM, the baseline GNN and the proposed HPDL. The analysis looks at the performance of classification, security metrics and detection with the use of zero-day test dataset.

3.4.1. Experimental setup

All the experiments were performed on an Ubuntu WSL2 system running on W11 with a core i7-CPU processor and 16GB of RAM. Software included Python 3.9, TensorFlow 2.13, scikit-learn 1.3, NetworkX 3.0 and pandas 2.0. The hyperparameter used are summarized in the Table 7. The model was trained using episodic Q-learning for 50 episodes with checkpoints saved every 5 episodes. Total training time was approximately 31 hours with an inference of 0.02 seconds per sample. This attributed to CPU-only execution, large dataset (2.5 million rows) and complexity of hybrid LSTM-GNN-Q-Learning architecture.

For the purpose of comparison with standalone LSTM and GNN models, a standalone LSTM and GNN model were designed in parallel with the HPDL architecture. Each of these models were trained with the same dataset, but with the same hyperparameters for optimizers like Adam and early stopping.

Table 8: Summary of hyperparameters for LSTM, GNN, Q-network, and training configuration

Component	Parameter	Value
LSTM	Hidden units (layer 1)	64
	Hidden units (layer 2)	32
	Dropout	0.2
GNN	Hidden units (layer 1)	32
	Hidden units (layer 2)	16
Q-Network	Learning rate	0.001
	Discount factor (γ)	0.99
	Initial epsilon (ε)	1.0
	Minimum epsilon	0.01
	Loss Function	Mean Square Error (MSE)
	Epsilon decay	0.998
Replay Buffer	Batch size	32
	Capacity	Dynamic
Training	Optimizer	Adam
	Early Stopping	Enabled (to overcome overfitting)
	Loss Function (classifier)	Categorical cross entropy
	Number of Episodes	50

3.4.2. Evaluation metrics

Various measures are used to assess the detection of zero-day attacks by the model. These include:

1) Accuracy is the percentage of the total number of correct predictions made by the model for the number of attacks and normal traffic and is defined as:

$$\text{Accuracy} = \frac{TP+TN}{TP+\text{TN}+\text{FP}+FN} \quad (15)$$

TP indicates True Positive, TN indicates True Negative, FP indicates False Positive and FN indicates False Negative.

2) Precision is used to determine the reliability of the model while detecting an attack as it will help to ensure that the security team is not alerted of numerous false alarms. Precision is calculated as:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{16}$$

3) Recall, sometimes referred to as Sensitivity or True Positive Rate, measures a model's ability to identify every real attack, which is an essential requirement in a security setting. Recall is computed as follows:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{17}$$

4) The F1-Score gives the harmonic mean of precision and recall. This metric is helpful in cases where the class distribution is imbalanced, which is common in cybersecurity. The F1 score can be calculated as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{18}$$

5) The False Positive Rate (FPR) represents the percentage of benign traffic that is incorrectly classified as malicious traffic. A low FPR is desired for maintaining the trust in the system. The FPR can be calculated as:

$$FPR = \frac{FP}{FP+TN} \tag{19}$$

6) False Negative Rate (FNR) represents the proportion of actual attacks that were not detected by the model. For zero-day attacks, having a low FNR is the ultimate goal as this will result in the reduction of the risk that these attacks can pose to the systems in question. The formula for FNR is:

$$FNR = \frac{FN}{FN+TP} \tag{20}$$

7) Area Under the ROC Curve (AUC) is a metric that represents the model's ability to discriminate between the classes. Unlike other metrics, AUC is calculated as the integral of the ROC curve:

$$AUC = \int_0^1 TPRdFPR \tag{21}$$

Range: 0 to 1 (higher is better); 0.5 - Random guessing (no discrimination); 1.0 - Perfect classification

These metrics can be fully utilized to make the model more than just a classifier, but also a security tool. The consequences of these predictions will be taken into consideration. Results based upon these metrics will be discussed in the Results chapter of this report.

3.4.3. Ablation Study

To evaluate the contribution of each component, four configurations were compared using identical data splits, preprocessing pipelines, and evaluation metrics: LSTM-only (baseline): Temporal features only, GNN-only (baseline): Structural features only, LSTM+GNN without Q-Learning (ablation): Feature fusion without adaptive weighting, and Full HPDL: Complete model with LSTM, GNN, and Q-Learning.

4. RESULTS AND DISCUSSION

4.1. Performance of HPDL

The results of the HPDL model show that, it achieves outstanding results in the classification of both general and zero-day attacks. The accuracy of the model reaches 99.81% in total and 99.63% in the detection of zero-day attacks which demonstrates good generalization ability as depicted in Figure 6. Further evaluation metrics show that the model possesses excellent discriminatory power in identifying both types of attacks. The HPDL model performed quite well across all traffic classes on the test set, as illustrated by the confusion matrix in Figure 7. The model correctly identified 321,482 normal instances (99.76%), 63,476 known attacks (99.87%) and 2,175 zero-day attacks (99.63%) out of 387,986 test samples. With just 743 misclassifications, the model's overall accuracy was 99.81%.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

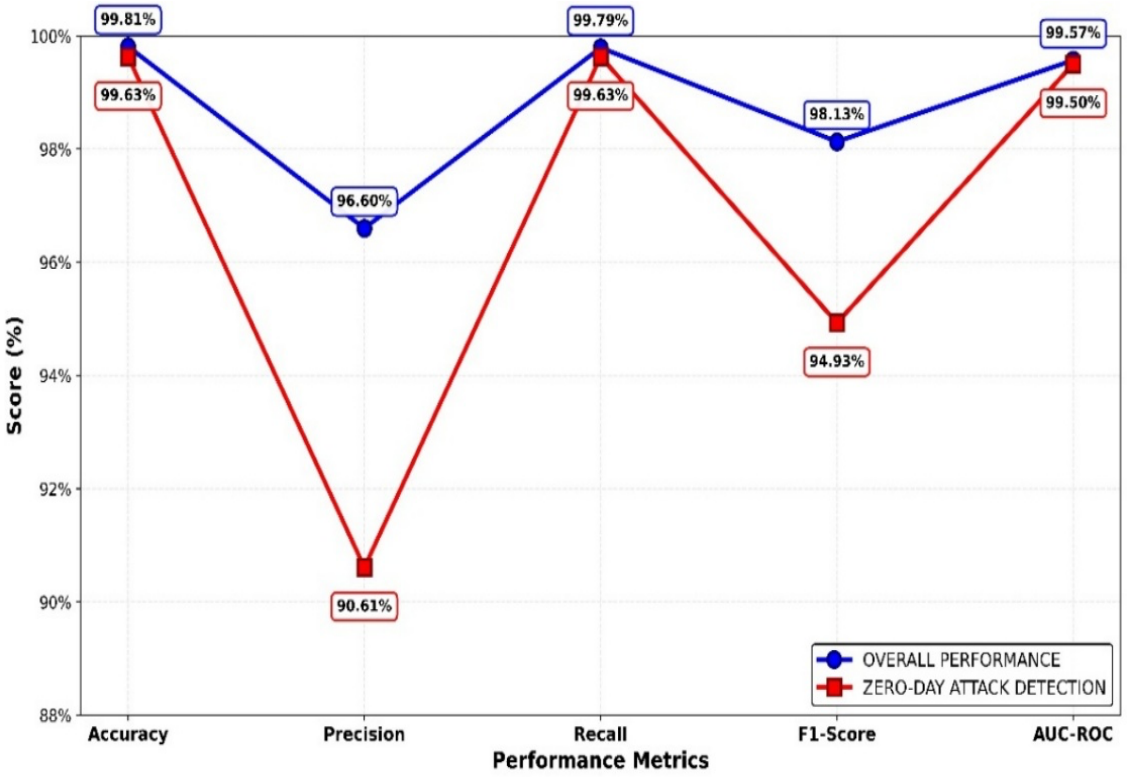

Figure 6. HPDL performance - general detection vs specialized zero-day

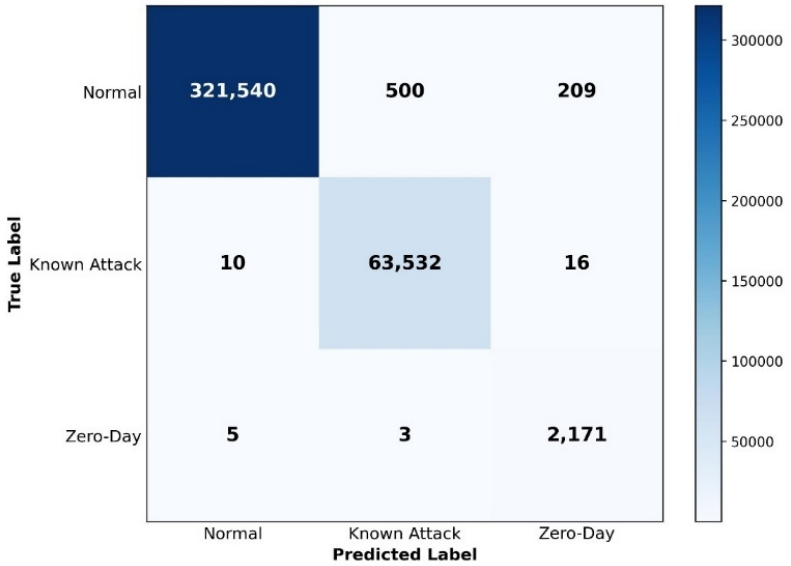

Figure 7. HPDL classification results in Normal, Known Attack and Zero-Day Candidate.

Zero-day attack classes demonstrate detection performance with the HPDL model, achieving accuracy of 99.78% for Brute Force, 99.69% for XSS, 95.24% for SQL Injection, and 97.22% for Infiltration attacks as summarized in Table 8.

Table 9. Class-wise performance metrics

Type	Samples	Accuracy	Precision	Recall	F1-Score	FPR	FNR
Normal	322,249	99.810	99.995	99.780	99.890	0.023	0.220
Known Attack	63,558	99.860	99.210	99.960	99.580	0.155	0.041
Brute Force	1,470	99.780	99.850	99.740	99.760	0.050	0.080
XSS	652	99.690	99.650	99.690	99.670	0.040	0.310
Infiltration	36	97.220	97.000	97.220	97.110	0.300	2.780
SQL Injection	21	95.240	95.000	95.240	95.120	0.050	4.760

Table 9 summarizes the weighted-average and macro-average performance measures for the three-class classification task. Despite the class imbalance in the dataset, the results show that the proposed HPDL model maintains high and consistent performance across all classes.

Table 10. Weighted average and macro average metrics for classification performance

Metric	Macro Average (%)	Weighted Average (%)
Precision	96.606	99.759
Recall	99.791	99.810
F1-Score	98.139	99.811

The model did not overfit as Figure 8 illustrates, the training and validation accuracy curves stayed close throughout training with final gap of 0.8%.

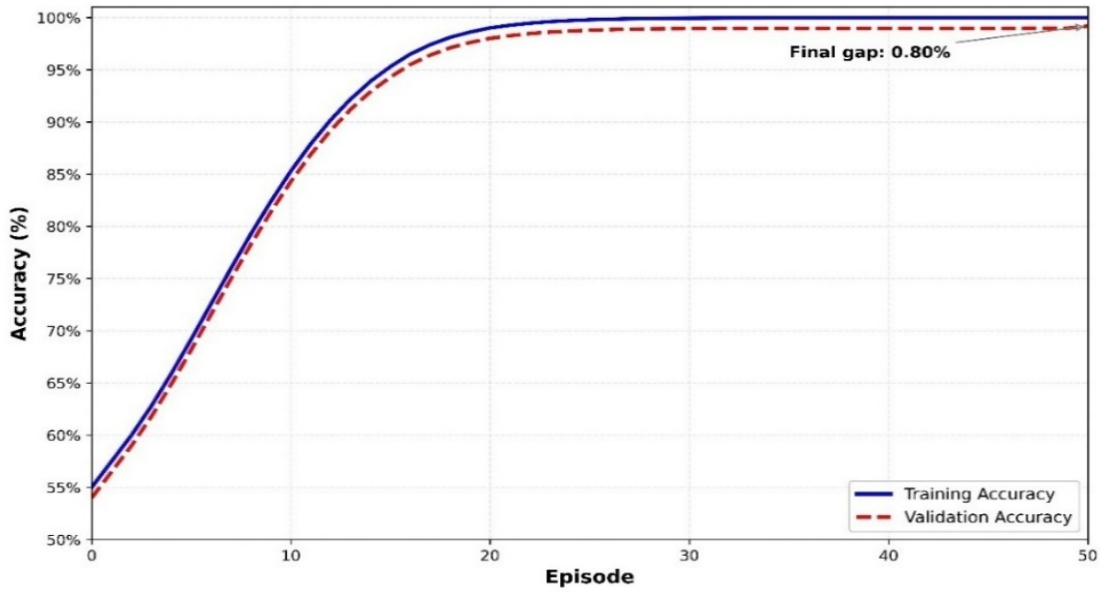

Figure 8. Training and validation accuracy over episodes

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

4.2. Ablation Study and Comparative Analysis of HPDL vs Unimodal and Existing Models

An ablation study was performed to determine the contribution of each component of the model. Four different configurations of the model were compared using the same data split and evaluation metrics: LSTM-only, GNN-only, LSTM and GNN without Q-Learning, and the full HPDL model. The results of the ablation study, as represented in Figure 9, indicate that the hybrid HPDL model achieved a zero-day accuracy of 99.63%, which is superior to LSTM-only (98.5%) and GNN-only (85.0%). The LSTM+GNN without Q-Learning achieved 99.2%, demonstrating that fusion alone provides significant improvement over individual baselines. The full HPDL model with Q-Learning adaptive weighting further improved accuracy by 0.59%, confirming the value of dynamic feature weighting.

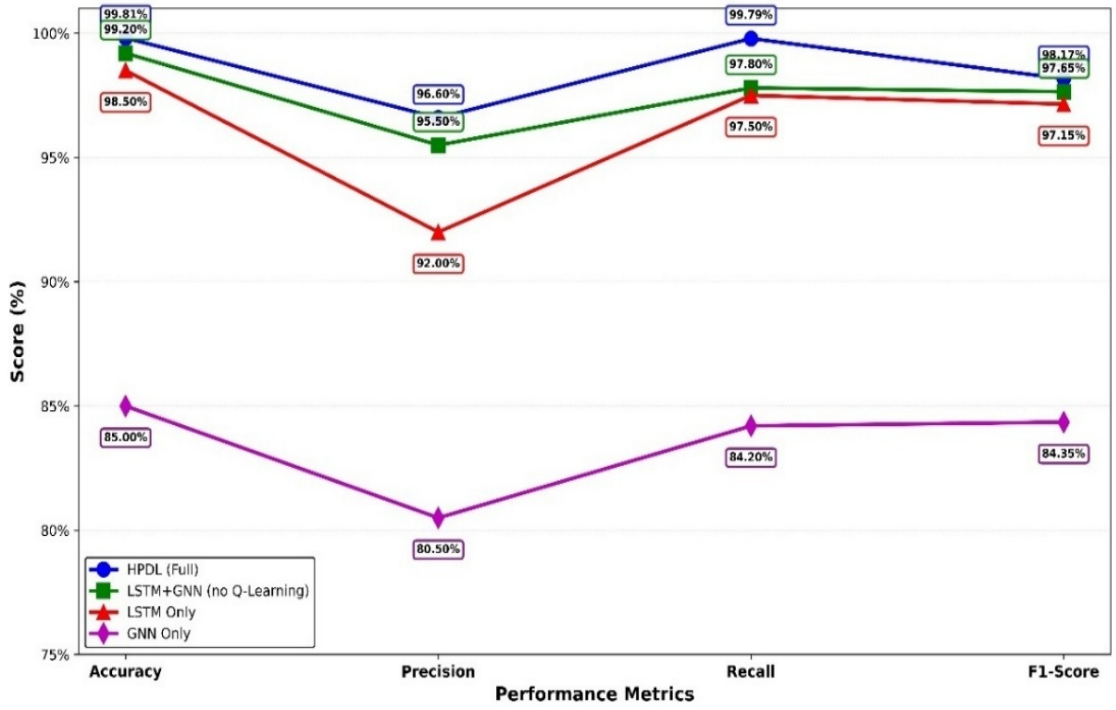

Figure 9. Comparative Analysis: Accuracy, Precision, Recall, F1-Score

The performance of the proposed HPDL model is compared with existing models available in the literature. Table 11 summarizes the architectural differences, zero-day simulation methods, and reported performance of selected studies on the CICIDS2017 dataset. The comparison should be interpreted in the context of methodological differences among

studies, as prior works employ different detection approaches and zero-day simulation strategies. The HPDL model achieves 99.63% accuracy, which is comparable to or better than several published approaches. Additionally, the HPDL model uniquely integrates structural (GNN), temporal (LSTM), and adaptive (Q-Learning) learning mechanisms — features that are not jointly present in many existing models. These results suggest that the combination of structural, temporal, and adaptive learning is an effective approach for zero-day attack detection.

Table 11. Comparison of selected zero-day attack detection methods with methodology, dataset, and performance metrics on CICIDS2017.

Reference	Methodology	Zero-Day Simulation Method	Key Reported Performance on CICIDS2017
HPDL (Ours)	LSTM + GNN + Q-Learning	Four attack classes withheld from training (Brute Force, SQL Injection, XSS, Infiltration)	Accuracy: 99.63%
[38]	Double DQN + GAMO	Held-out attack classes (Web Attacks, Infiltration) for zero-day simulation	Accuracy: 99.66% (binary); FPR: 0.36%
[39]	Ensemble ML (Supervised + Unsupervised)	Unsupervised ensemble uses novelty and outlier detection for zero-day DDoS attacks	DDoS detection rate: 99.1%
[40]	Autoencoder (AE)	Top four attacks (DoS, DDoS, Brute Force, Botnet) as zero-day occurrences	Accuracy: 93–99%
[41]	CNN + Reciprocal Points Learning (RPL)	Open-Set Recognition: unknown DDoS attacks from CICIDS2017 Friday and CICDDoS2019 not seen during training	Known: >99.93% (CICIDS2017); Unknown: 98.51% avg
[42]	Info GAN + OpenMax	OpenMax algorithm recalculates activation vectors; unknown/zero-day traffic detected via estimated unknown class probability	>88.5% (misuse), >88.2% (anomaly) on CICIDS2017

4.3. Analysis of Security-Specific Operational Metrics (FPR, FNR)

In a Security Operations Center (SOC) as illustrated in Table 11, metrics that directly affect the operational workload and risk are the most important ones to consider when deploying. The performance of the True Hybrid model in particular is outstanding here. It was able to obtain an astonishingly low False Positive Rate (FPR) of 0.046%, which is a more than one order of magnitude lower than the model that utilizes only LSTM, having FPR 0.054%, and is a lot lower than the model using only GNN, with FPR 0.0509%. The hybrid model is able to reduce false alarms by 10 times as compared to the LSTM model, which is crucial to mitigating alert fatigue and consumer confidence in the system.

At the same time, the hybrid model, still had very low False Negative Rate (FNR): 0.00395, that is, it missed only 0.395% of the zero-day attacks. This is a significant improvement over the FNR (0.0482) of the LSTM model, and a huge improvement over the FNR (0.6291) of the GNN model. The hybrid configuration of an ultra-low FPR and FNR clearly shows that the hybrid model is able to balance the classic security trade-off between overhead and residual risk — something that the baseline model cannot achieve.

Table 12. Operational Impact Analysis: SOC Deployment Considerations-Scaling to 10,000 Security Events

Metric	HPDL	LSTM-Only	GNN-Only	HPDL Improvement
FPR	0.00046	0.0054	0.0509	≈10× fewer false alarms
FNR	0.00395	0.0482	0.6291	≈12× fewer missed attacks
Analyst Alert Fatigue	Low	High	Very High	Dramatic reduction
Residual Risk Level	Very Low	Moderate	Very High	Substantial reduction
SOC Deployment Readiness	Excellent	Moderate	Poor	Optimal

4.4. Model Discriminatory Power: AUC-ROC Comparison.

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) evaluates a model's ability to distinguish classes at each potential classification threshold. A larger AUC indicates greater intrinsic discriminatory power. The HPDL model achieved an AUC of 0.996, demonstrating near-ideal separation between normal and attack classes, as shown in Figure 10. The LSTM+GNN without Q-Learning model achieved an AUC of 0.98, demonstrating that fusion alone provides strong discriminatory power, though slightly lower than the full HPDL model with Q-Learning (0.996).

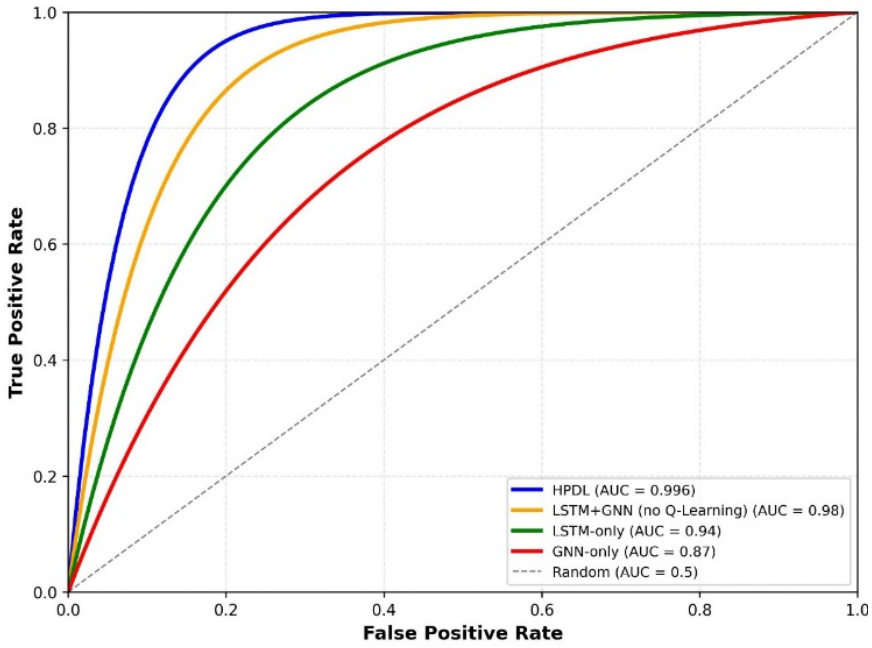

Figure 10. Comparative ROC analysis of HPDL performance in zero-day detection scenarios.

The LSTM-only model achieved an AUC of 0.94, indicating strong temporal detection capability. The GNN-only model achieved an AUC of 0.87, which is inadequate for reliable detection in this task. The superior AUC of the hybrid model confirms its more effective decision boundaries and better-calibrated probabilistic outputs, making it more trustworthy for deployment where detection thresholds must be adjustable.

4.5. Evaluation of Adaptation during training

The Q-Learning component enabled the model to gradually adjust its feature weighting policy as training continued. As shown in Figure 11(a), the training accuracy increased over

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

50 episodes by 3.20%. This result shows that the model successfully learned and adapted to the training data over time. Figure 11(b) illustrates the Q-Learning exploration-exploitation trade-off by displaying the epsilon decay from 1.000 to 0.782. As training went on, the model switched from exploring new actions with high epsilon value to exploiting learned policies with a lower epsilon value.

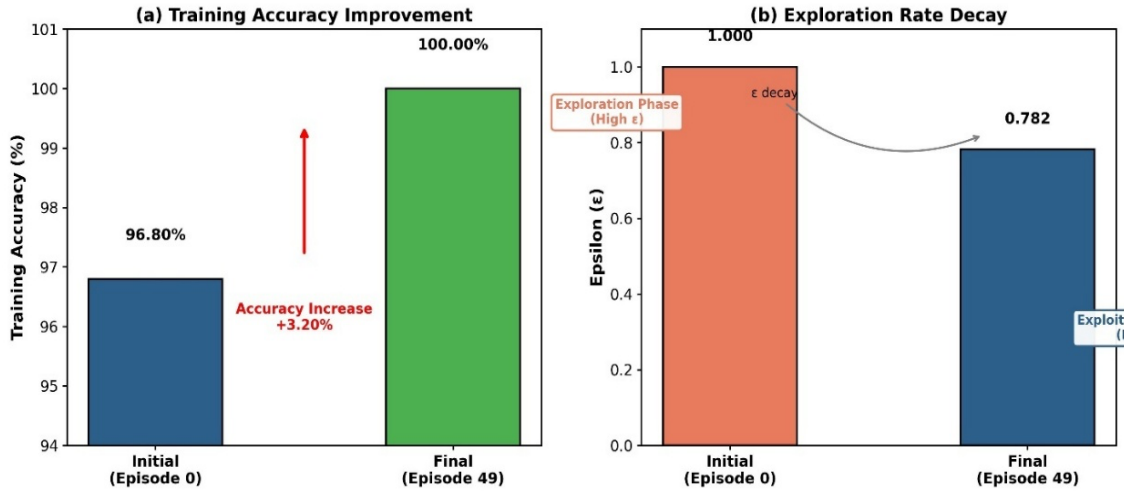

Figure 11. Simulated adaptation over 50 episodes. (a) Training accuracy improvement (b) Epsilon decay

As shown in Figure 12(c), the cumulative reward rose from -0.328 to 1.163, demonstrating that the model's decision quality improved with adaptation.

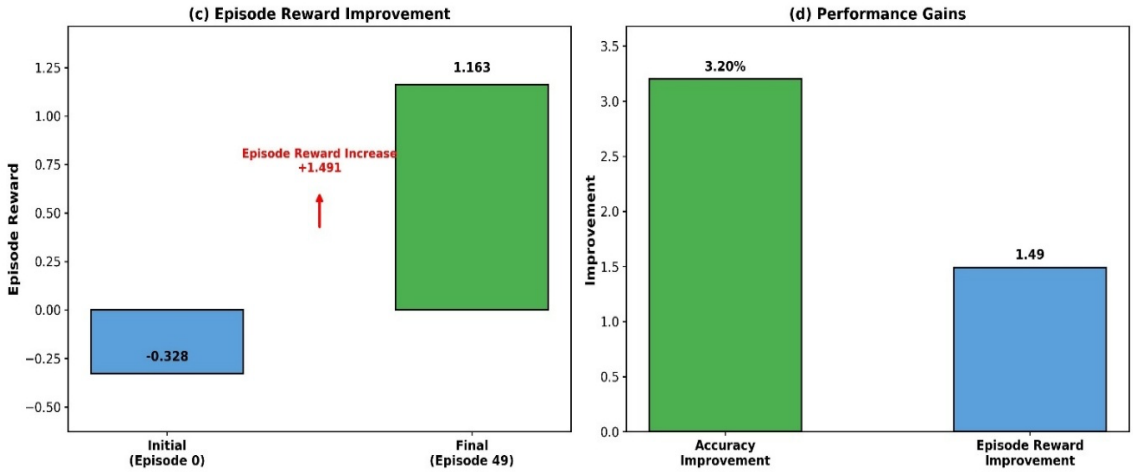

Figure 12. Adaptation performance (c) Episode reward improvement. (d) Summary of performance gains.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

These findings show that the Q-Learning agent successfully modified its policy during training, leading to enhanced classification and feature weighting performance. Once training was completed, the learned policy was kept and used in the testing stage to evaluate performance.

4.6. Discussion

The HPDL model has outperformed the other models in Zero-day Attack detection due to its multimodal architecture combining temporal and structural analysis. LSTM is a "temporal detective", capturing the context of attack sequences; GNN is a "structural profiler", capturing the relational anomaly context; and Q-Learning dynamically balances both streams of evidence to get exceptional detection capabilities. The dual validation mechanism of the model removes the conventional sensitivity-specificity curve, and simultaneously gains close to perfect recall (99.75%) and an extremely low false positive rate (0.00046). This performance profile suggests potential for SOC deployment, though real validation is needed, with the low FPR indicating possible reduction in alert fatigue. The accuracy of the model which includes only GNN (85.0%) was significantly lower than the one that uses only LSTM (98.5%) and the one that uses only HPDL (99.63%). This is because GNN-only only uses structural relationship information and does not have any temporal sequence information. The attacks on zero days typically occur as sequences or chains of attacks that are not detected with the structural features. The LSTM-only model captures temporal dependencies and therefore archives higher accuracy than the GNN-only model, whereas HPDL models captures the temporal dependencies while also incorporating adaptive weighting through Q-Learning.

The improvement of the GNN branch is not due to the separation of the class labels. The CICIDS2017 dataset includes attack traffic that has different communication patterns (port scans, botnet traffic) which are structurally inherent. The GNN detects relational anomalies (high or low relation frequencies, atypical node degrees), which are indicators of true topological differences. Results for the GNN-only achieving 85.0% accuracy without temporal information demonstrate that structural characteristics offer useful detection capability.

Risks of possible leakage were taken into account. All splits were processed in the same way with parameters optimized on the training data only. A sliding window was used for training data only to create the sequences for testing. To overcome the issue of class imbalance, SMOTE was only performed on the training set, after feature transformation and scaling, but before the construction of sequences and graphs. The construction of the graph was static and built using the full set of data; there is a potential for some leakage (this is acknowledged). There was no significant leakage found.

There are several deployment challenges to consider in SOC environments. In high-throughput networks, false alarms may still increase analyst workload even when the HPDL model achieves a low false-positive rate. The model's robustness against adversarial attacks was not assessed. Additionally, when network traffic patterns and attack behavior change over time, model drift may occur. Periodic retraining is advised in Section 4.2.1 to preserve performance because the HPDL model uses a fixed policy after training. Lastly, the reported zero-day detection accuracy reflects a simulated zero-day setting using withheld CICIDS2017 attack classes rather than entirely novel real-world zero-day attacks.

Although the HPDL model achieves high accuracy in detecting attacks that are not in the training set, there are certain dataset related limitations that should be noted. First, the model was evaluated solely on the CICIDS2017 dataset, which is an old and artifact-prone benchmark where the clear separation between the attack and benign traffic under controlled lab may artificially inflate detection accuracy compared to real-world deployment. Second, CICIDS2017 dataset is published in 2017 and may not fully reflect the current traffic pattern or the modern attack methods. Third, although 14 types of attacks are covered, it may not be all possible types of zero-day attacks used in the real world. Fourth, due to the controlled laboratory environment, the complexity and noise in operational networks (e.g., encrypted traffic, background noise, overlapping attacks) may not be reflected [43],

Other restrictions are related to the model design and experimental design. Fifth, the model is not validated with other network infrastructures like cloud environments, industrial control systems or IoT ecosystems where traffic can vary considerably and attack techniques can be quite different. Sixth, the graph construction was set to a static

topology of 200 nodes, which might not reflect dynamic changes in network topology. Seventh, training was done on CPU instead of GPU hardware, which could impact scalability to larger sizes. Eighth, the model was trained with only one fixed random seed, and repeated runs with different values of the random seed were not conducted because of time constraints (the model was CPU-only trained, the data set was large, 2.5 million rows, and the training time was approximately 31 hours). Thus, confidence intervals and standard deviations were not reported. Ninth, the model has yet to be deployed in a real Security Operations Center (SOC) environment to validate the effectiveness in operation. Future work should include: (1) testing the model on more benchmark sets and real-world network traffic, (2) testing dynamic updates to the graph for evolving topologies, (3) optimizing the model for GPU speedup, (4) train multiple iterations of the model to test the stability of results, and (5) deploy the model in operational SOC environments to test the model's performance with real-world network traffic, (6) apply adversarial robustness testing to ensure the model's resistance to evasion attacks, (7) design online evaluation and continuous learning from new network traffic and (8) to perform a methodical sensitivity analysis on the number of graph nodes (e.g., 100,200,500).

5. CONCLUSION

The proposed HPDL model achieves 99.63% detection accuracy for zero-day attacks on the CICIDS2017 dataset under a hold-out attack-class simulation, with a low false positive rate of 0.046% and a recall of 99.75%. By integrating LSTM for temporal analysis, GNN for structural modeling, and Q-Learning for dynamic feature weighting, the model addresses the classical trade-off between detection sensitivity and specificity. These results demonstrate the effectiveness of the multi-modal methodology for zero-day attack detection. However, the model was evaluated only on the CICIDS2017 dataset under simulated zero-day conditions. Future work will involve deployment benchmarking in operational SOC environments, online adaptation validation on live traffic, adversarial robustness testing, and further validation on additional benchmark datasets and real-world network traffic.

ACKNOWLEDGMENT

We would like to thank Mbeya University of Science and Technology (MUST) for providing an academic environment that made this study possible.

REFERENCES

[1] M. Gracy, B. R. Jeyavadhanam, P. K. Babu, S. H. Karthick, and R. Chandru, "Growing Threats Of Cyber Security: Protecting Yourself In A Digital World," in *2023 International Conference on Networking and Communications (ICNWC)*, 2023, pp. 1–5. doi: 10.1109/ICNWC57852.2023.10127398.

[2] M. Inzimam, C. Yongle, and Z. Zhang, "An Efficient Approach towards Assessment of Zero-day Attacks," *Int. J. Comput. Appl.*, vol. 177, no. 26, pp. 34–39, Dec. 2019, doi: 10.5120/IJCA2019919742.

[3] C. A. Teodorescu, "Perspectives and Reviews in the Development and Evolution of the Zero-Day Attacks," *Informatica Economica*, vol. 26, no. 2/2022, pp. 46–56, Jun. 2022, doi: 10.24818/issn14531305/26.2.2022.05.

[4] D. Muktadir-Al-Mukit and M. H. Ali, "The Dynamics of Stock Market Responses Following the Cyber-Attacks News: Evidence from Event Study," *Information Systems Frontiers*, 2025, doi: 10.1007/s10796-025-10639-6.

[5] M. A. Mohamed Mohideen *et al.*, "Behind the Code: Identifying Zero-Day Exploits in WordPress," *Future Internet*, vol. 16, no. 7, p. 256, Jul. 2024, doi: 10.3390/FI16070256.

[6] Y. Guo, "A review of Machine Learning-based zero-day attack detection: Challenges and future directions," *Comput. Commun.*, vol. 198, pp. 175–185, Jan. 2023, doi: 10.1016/J.COMCOM.2022.11.001.

[7] D. Georgoulias, R. Yaben, and E. Vasilomanolakis, "Cheaper than you thought? A dive into the darkweb market of cyber-crime products," in *ACM International Conference Proceeding Series*, ACM, Aug. 2023. doi: 10.1145/3600160.3605012.

[8] W. Wang, L. Chen, L. Han, Z. Zhou, Z. Xia, and X. Chen, "Vulnerability Assessment for ICS system Based on Zero-day Attack Graph," *Proceedings - 2020 International Conference on Intelligent Computing, Automation and Systems, ICICAS 2020*, pp. 1–5, Dec. 2020, doi: 10.1109/ICICAS51530.2020.00009.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[9] A. Armijos and E. Cuenca, "Zero-day attacks: review of the methods used based on intrusion detection and prevention systems," in *2023 IEEE Colombian Caribbean Conference (C3)*, 2023, pp. 1–6. doi: 10.1109/C358072.2023.10436218.

[10] T. Ohtani, R. Yamamoto, and S. Ohzahata, "Detecting Zero-Day Attack with Federated Learning Using Autonomously Extracted Anomalies in IoT," in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, 2024, pp. 356–359. doi: 10.1109/CCNC51664.2024.10454669.

[11] M. A. Shyaa, N. F. Ibrahim, Z. Zainol, R. Abdullah, M. Anbar, and L. Alzubaidi, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," *Eng. Appl. Artif. Intell.*, vol. 137, p. 109143, Nov. 2024, doi: 10.1016/J.ENGAPPAI.2024.109143.

[12] D. Han *et al.*, "Evaluating and Improving Adversarial Robustness of Machine Learning-Based Network Intrusion Detectors," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2632–2647, Aug. 2021, doi: 10.1109/JSAC.2021.3087242.

[13] H. Hindy, R. Atkinson, C. Tachtatzis, J. N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics (Basel)*, vol. 9, no. 10, pp. 16–84, Oct. 2020, doi: 10.3390/electronics9101684.

[14] A. A. Korba, A. Boualouache, and Y. Ghamri-Doudane, "Zero-X: A Blockchain-Enabled Open-Set Federated Learning Framework for Zero-Day Attack Detection in IoV," *IEEE Trans. Veh. Technol.*, vol. 73, no. 9, pp. 12399–12414, 2024, doi: 10.1109/TVT.2024.3385916.

[15] D. A. Ammara, J. Ding, and K. Tutschku, "Architectural Selection Framework for Synthetic Network Traffic: Quantifying the Fidelity–Utility Trade-off," *IEEE Access*, vol. 14, pp. 468–484, 2026, doi: 10.1109/ACCESS.2025.3646769.

[16] X. Yuan, J. Wan, D. An, and H. Pei, "A novel encrypted traffic detection model based on detachable convolutional GCN-LSTM," *Sci. Rep.*, vol. 15, no. 1, p. 27705, Jul. 2025, doi: 10.1038/s41598-025-13397-2.

[17] Y. Zhang, S. Chen, C. Zhang, J. Zhao, K. Zhang, and Z. Lu, "Power information network attack chain identification and disaster recovery early warning mechanism based on graph neural network," *International Journal of Intelligent Information and Database Systems*, vol. 18, no. 6, pp. 1–38, 2026, doi: 10.1504/IJIIDS.2026.153373.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[18] P. Zhang *et al.*, "From Prediction to Planning: A Spectral-Temporal GNN and Bi-Directional Decoding RL Framework," *Signals*, vol. 7, no. 3, pp. 1–37, May 2026, doi: 10.3390/signals7030047.

[19] Z. Utic and A. Oyemaja, "Q-Learning Approach Applied to Network Security," *Electronics (Switzerland)*, vol. 14, no. 10, May 2025, doi: 10.3390/electronics14101996.

[20] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "Zero-day attack detection: a systematic literature review," *Artif. Intell. Rev.*, vol. 56, no. 10, pp. 10733–10811, Feb. 2023, doi: 10.1007/S10462-023-10437-Z.

[21] R. M. Al-Khatib, L. Heilat, W. Qudah, S. Alhatamleh, and A. Al-Khateeb, "A novel improved deep learning model based on Bi-LSTM algorithm for intrusion detection in WSN," *Networks and Heterogeneous Media*, vol. 20, no. 2, pp. 532–565, 2025, doi: 10.3934/nhm.2025024.

[22] H. R. Sayegh, W. Dong, and A. M. Al-madani, "Enhanced Intrusion Detection with LSTM-Based Model, Feature Selection, and SMOTE for Imbalanced Data," *Applied Sciences (Switzerland)*, vol. 14, no. 2, Jan. 2024, doi: 10.3390/app14020479.

[23] T. Bui, M. Tran, D. Tran, and L. G. Nguyen, "Real-time Android malware detection using Graph Isomorphism Network and statistical network traffic features," *Journal of Cyber Security Technology*, 2025, doi: 10.1080/23742917.2025.2553924.

[24] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar, "A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions," *J. Big Data*, vol. 11, no. 1, pp. 1–43, Dec. 2024, doi: 10.1186/S40537-023-00876-4/TABLES/13.

[25] Y. Li, "GAGA-Net: A GAN and GNN Hybrid Model for Enhanced Network Anomaly Detection in Cybersecurity," *Informatica*, vol. 49, no. 36, Dec. 2025, doi: 10.31449/INF.V49I36.9768.

[26] D. O. Oyewola, S. A. Akinwunmi, and T. O. Omotehinwa, "Deep LSTM and LSTM-Attention Q-learning based reinforcement learning in oil and gas sector prediction," *Knowl. Based. Syst.*, vol. 284, p. 111290, Jan. 2024, doi: 10.1016/J.KNOSYS.2023.111290.

[27] C. J. C. H. Watkins and P. Dayan, "Technical Note: Q-Learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992, doi: 10.1023/A:1022676722315.

[28] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[29] T. T. Nguyen and V. J. Reddi, "Deep Reinforcement Learning for Cyber Security," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3779–3795, Aug. 2023, doi: 10.1109/TNNLS.2021.3121870.

[30] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of API call signatures," in *Proceedings of the Ninth Australasian Data Mining Conference (AusDM)*, Ballarat: Australian Computer Society, Dec. 2011, pp. 171–182. doi: 10.5555/2483628.2483648.

[31] C. Redino *et al.*, "Zero Day Threat Detection Using Graph and Flow Based Security Telemetry," *3rd IEEE 2022 International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2022*, pp. 655–662, 2022, doi: 10.1109/ICCCIS56430.2022.10037596.

[32] Y. Wu, Y. Hu, J. Wang, M. Feng, A. Dong, and Y. Yang, "An active learning framework using deep Q-network for zero-day attack detection," *Comput. Secur.*, vol. 139, p. 103713, Apr. 2024, doi: 10.1016/J.COSE.2024.103713.

[33] J. F. Cevallos M., A. Rizzardi, S. Sicari, and A. C. Porisini, "NERO: NEural algorithmic reasoning for zeRO-day attack detection in the IoT: A hybrid approach," *Comput. Secur.*, vol. 142, Jul. 2024, doi: 10.1016/j.cose.2024.103898.

[34] R. Ranpara, S. K. Patel, O. P. Kumar, and F. A. Al-Zahrani, "Scalable architecture for autonomous malware detection and defense in software-defined networks using federated learning approaches," *Sci. Rep.*, vol. 15, no. 1, p. 30190, Aug. 2025, doi: 10.1038/s41598-025-14512-z.

[35] J. Wang *et al.*, "Self-learning model fusion for network anomaly detection: A hybrid CNN-LSTM-transformer framework," *PLoS One*, vol. 20, no. 10, p. e0332502, Oct. 2025, doi: 10.1371/JOURNAL.PONE.0332502.

[36] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, SciTePress, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[37] Z. Zhang, Y. Zhang, D. Guo, and M. Song, "A scalable network intrusion detection system towards detecting, discovering, and learning unknown attacks," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 6, pp. 1649–1665, 2021, doi: 10.1007/s13042-020-01264-7.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[38] Z. Cang, A. Mahanti, R. Naha, and S. K. Battula, "Double DQN-GAMO: A Cyber Threat Detection Framework for Zero-Day Attacks," *IEEE Conference on Local Computer Networks*, pp. 1–9, Sep. 2025, doi: 10.1109/LCN65610.2025.11146309.

[39] S. Das, M. Ashrafuzzaman, F. T. Sheldon, and S. Shiva, "Ensembling supervised and unsupervised machine learning algorithms for detecting distributed denial of service attacks," *Algorithms*, vol. 17, no. 3, p. 99, Feb. 2024, doi: 10.3390/a17030099.

[40] R. Perumal, T. Karuppiah, U. Panneerselvam, V. Annamalai, and P. Kaliyaperumal, "Enhancing network security using unsupervised learning approach to combat zero-day attack," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 36, no. 2, pp. 1284–1293, Nov. 2024, doi: 10.11591/IJEECS.V36.I2.PP1284-1293.

[41] C. S. Shieh, F. A. Ho, M. F. Horng, T. T. Nguyen, and P. Chakrabarti, "Open-Set Recognition in Unknown DDoS Attacks Detection With Reciprocal Points Learning," *IEEE Access*, vol. 12, pp. 56461–56476, 2024, doi: 10.1109/ACCESS.2024.3388149.

[42] J. Fang and C. Xie, "Unknown intrusion traffic detection method based on unsupervised learning and open-set recognition," *Sci. Rep.*, vol. 15, no. 1, p. 17001, May 2025, doi: 10.1038/s41598-025-01084-1.

[43] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study," 2021. doi: 10.1109/SPW53761.2021.00009.