



# INTELLIGENT NETWORK INTRUSION DETECTION USING FEA-CASSTFORMERNET OPTIMIZED BY ARCHERFISH HUNTING ALGORITHM

S. Tamil Selvan<sup>\*1</sup>, Haritha K Sivaraman<sup>2</sup>, T. Anupama Thankam<sup>3</sup>, Gavaskar Vincent<sup>4</sup>

<sup>1</sup>Associate Professor & Head, Department of Computer Science and Design, Erode Sengunthar Engineering College (Autonomous), Thudupathi, Perundurai, Erode, Tamil Nadu, India.

<sup>2</sup>Associate Professor, Department of Electronics and Communication Engineering, RajaRajeswari College of Engineering, Bengaluru, Karnataka, India.

<sup>3</sup>Department of Computer Science and Engineering, Noorul Islam Centre for Higher Education, Kumarakovil, Tamil Nadu, India.

<sup>4</sup>Professor, Department of Computer Science and Engineering, Vins Christian College of Engineering, Chunkankadai, Nagercoil, Tamil Nadu, India.

<sup>1</sup><https://orcid.org/0009-0004-8835-7342>, <sup>2</sup><https://orcid.org/0009-0003-1607-1962>,

<sup>3</sup><https://orcid.org/0009-0007-3984-0241>, <sup>4</sup><http://orcid.org/0009-0009-5891-7712>

Email: \*[stamilselvancse@gmail.com](mailto:stamilselvancse@gmail.com), [harithainhere@gmail.com](mailto:harithainhere@gmail.com), [anupthankam1976@gmail.com](mailto:anupthankam1976@gmail.com), [csevcwce@gmail.com](mailto:csevcwce@gmail.com)

## ARTICLE INFO

### Article History

Received: January 13, 2026

Reviewed: February 7, 2026

Accepted: April 17, 2026

Published: May 29, 2026

### Keywords:

IDS,  
Preprocessing,  
Exploratory Data Analysis,  
Feature Engineering,  
AfHO- FEA-CasSTFormerNet.

## ABSTRACT

Through the rapid growth in complexity of cyber-attacks on networked platforms, there is a need for advanced and adaptable Intrusion Detection Systems (IDSs) featuring real-time analysis and reliable categorization capabilities. This paper presents an enhanced hybrid approach to IDS based on sophisticated Deep Learning (DL) architectures. It discusses with some of the procedures which involves the following stages. First, the process begins by collecting all of the unprocessed data collected from network log records. Next, each data field goes through an extensive pre-processing phase that addresses not only missing values via a KNN Imputer, also includes Min-Max scaling to normalize the data as well as Hybrid encoding of categorical variables (one-hot encoding and entity embedding) to create appropriately formatted data by the DL algorithms. Further analysis of class imbalances is conducted using Exploratory Data Analysis (EDA) methods that takes both distributional data through pie-chart views and correlation assessments comparing all of the different fields of the dataset using a heatmap format. Features are extracted using Shannon Entropy for purposes of meaningful representation learning, while dimensionality reduction and covariance reduction will be accomplished through the use of Principal Component Analysis (PCA). To ensure a fair evaluation, the dataset used for testing is split in half, one for training and the other for testing. The framework suggested is FEA-CasSTFormerNet, this framework combines both recursive attention through memory mechanisms using Fractal Echo Attention (FEA) and deep temporal and spatial relationship modeling using Cascaded Spatio Temporal Transform (CasSTFormer). To balance exploration versus exploitation, the framework utilizes Archerfish Hunting Optimization (AfHO) to adjust hyper-parameters during the archerfish's behavior of chasing prey. Through experimentation with benchmark datasets for IDS inscribed in Python, results demonstrate that the proposed framework provides extremely high accuracy of 0.9998, precision of 0.9988, Recall of 0.9960 and FI-Score of 0.9974 at identifying diverse attack classes, less false positives, and greater generalization ability than existing solutions.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

## I. INTRODUCTION

Cybersecurity has become a major issue throughout the world with an impact on the security of people, businesses, and countries. The rapid growth of technology has led to an increase in both the number and sophistication of network attacks [1]. During the past several years, many researchers have focused on DL algorithms to improve the ability to detect malicious activity.

Network security threats are not only diverse, but they also have an increasing level of severity. Threats come from a variety of sources, including traditional viruses and malware, advanced phishing schemes, ransomware, and zero-day vulnerabilities. While traditional Denial of Service (DoS) attacks and unauthorized access continue to threaten organizations, newer methods of cybercrime such as data breaches and identity theft are becoming more prevalent [2-4]. The environment in which networks currently operate continues to become more complicated, and at the same time, there are constantly developing attack vectors that make standard cyber defense technologies increasingly difficult for protecting against unknown attacks. The traditional methods of using rule-based and signature-based detection to monitor network traffic for signs of IDS have become increasingly inadequate for detecting unknown intrusions [5], [6]. The centralized server and the participating nodes of an FL based IDS operate within a systematic architecture and do not exchange real data. Instead, real-time model updates are provided between the two communicating parties. This aspect of FL provides it with a unique set of capabilities that makes it a better option than traditional AI algorithms where the node users share real data with the centralized server. To achieve the goal of intruder detection, the FL system begins by sending out a basic or initial global model from the centralized server to the nodes involved in the communication. Then the initial model will be trained and analysed using the locally generated data that individual nodes have stored [7-9]. Once each node has completed training, all nodes will transmit updates about how the individual node's trained model has been modified to the centralized server. The updated models will be securely compiled into a single global model using the Multi-Party Secure Aggregation (MPSA) method at the centralized server. Finally, the FL system sends an updated global model to the subordinate nodes so that it may be used by the subordinate nodes for detecting intruders in their respective local environments [10-12].

The threat of cyber-attacks has led to an increased emphasis on improving network and computer system security. To this end, most modern security technologies rely on defense in Depth strategies, which leverage multiple security methods. Specifically, Network Intrusion Detection Systems (NIDS) appear to be highly effective at detecting a wide range of anomalous activity as compared to traditional Host-Based IDS, because of the high degree of flexibility that NIDS provide [13], [14]. The benefit of NIDS is that they can detect and/or prevent attacks before they reach a host by monitoring network traffic. In addition to their efficiency, NIDS also have the ability to provide internal and external protection across multiple layers (services, devices and architectures). NIDS have the added advantage of providing protection for all devices needing protection without requiring each device on the network to deploy an IDS. Rather than requiring an IDS to be deployed on every device, NIDS only need to have access to the applicable network traffic [15]. In general, however, the current NIDS solutions based on expert knowledge require constant updating to stay relevant against new threats, as well as continual management by an expert who can create rules for detecting attacks. Unfortunately, as technology continues to progress and change, the constant need for updates may adversely affect the performance of current NIDS and their ability to detect new types of attacks and/or increase the rates of false positives [16]. To tackle this issue, Machine Learning (ML) techniques are used, particularly Deep Neural Network (DNN), by virtue of its ability to effectively extract important features from data and accurately forecast future output has transformed many different sectors. This is evidenced by the advances that it has made in image classification, natural language processing, speech recognition, data generation, cybersecurity, digital forensics [17], [18]. Therefore, while DNNs have proven to be extremely successful in controlled circumstances, they are also very vulnerable to a number of targeted assault methods when used in an adversarial environment. Many methods to assault ML systems have emerged over the last few years, including both Training-time attacks (e.g., backdoor, poisoning) and Evasion attacks at Inference-time (e.g., adversarial samples) [19]. Evasion attacks specifically represent a critical hazard to NIDS because of the large amount of flexibility that an attacker has in modifying their traffic to avoid being detected [20]. To reduce the impact of this limitations, this study introduces a DL technique, which is FEA-CasSTFormerNet used to accurately detect the NIDS.

Table 1: Related Works of Existing Models for Detecting Nids.

| SI. NO | Author/Year                             | Models   | Merits                                                                                                                                                                                                                          | Demerits                                                                                                                                                                            | Performance metrics                                                                                  |
|--------|-----------------------------------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| 1.     | Himanshu Sharma et al [2024] [21]       | RNN      | RNNs have a great fit for IoT traffic because it is time-sensitive or time-based information and use temporal relationships to improve the accuracy of anomaly detection.                                                       | These models require many instances in their initial datasets to train well along with the fact that they can also cause delays in detecting anomalies as they happen in real-time. | It attains an accuracy, precision, recall and FI-score of 0.95, 0.96, 0.95 and 0.94                  |
| 2.     | Xiaoyun Ye et al [2025] [22]            | TCN      | It provides improved scalability, rapid training times, and enhanced accuracy for detecting slow insider threats                                                                                                                | The typical method for training a model using both technologies requires considerable amounts of quality log data.                                                                  | It achieves an accuracy of 0.90, precision of 0.87, recall of 0.93 and FI-score of 0.90              |
| 3.     | Fatimetou Abdou Vadhi et al [2024] [23] | Ensemble | By combining different models together, we have a more robust solution to the problems associated with noisy and imbalanced web traffic data.                                                                                   | The design, tuning, and maintenance of ensemble models are very complex because every model in the ensemble has its own unique hyperparameters.                                     | The performance of ensemble is 0.995 of accuracy, precision, recall and 0.994 of FI-score.           |
| 4.     | Nitu Dash et al [2025][24]              | PSO-LSTM | LSTMs are very useful when learning about the temporal correlation of network traffic.                                                                                                                                          | LSTMs requires substantial amounts of labeled data, this data can be prohibitively costly or not easily available.                                                                  | The metrics of PSO-LSTM IS 0.91 of accuracy, 0.85 of precision, 0.93 of recall and 0.90 of FI-score. |
| 5.     | Mohammad Shoab et al [2024] [25]        | GRU      | Due to their lightweight nature and low-computational requirements, GRUs are ideal for use on low-resource IoT devices. At the same time, GRUs are able to effectively capture the temporal characteristics of network traffic. | The system is too resource-intensive for ultra-low-power IoT nodes, therefore limiting the ability to deploy at the edge and instead deploying only at the gateway.                 | It achieves an accuracy of 0.998, precision of 0.991, recall of 0.987 and FI-score of 0.995          |

Source: Authors, (2026).

## I.1 CONTRIBUTIONS

- To improve the quality of data by preprocessing the dataset by normalising features, filling missing values and encoding categorical attributes.
- To identify correlation patterns and class distribution characteristics, EDA is implemented.
- To maximise the feature space using entropy-based features and to decrease dimensionality, PCA is used to achieve the best possible model performance.
- To effectively predict types of attacks, data-splitting method is introduced.

## II. PROPOSED SYSTEM DESCRIPTION

The proposed system for this paper is to create a hybrid intelligent and adaptive IDS that address the increasing issues with the complexity of cyber threats affecting modern-day networks. Figure 1 depicts the block diagram of the proposed model.

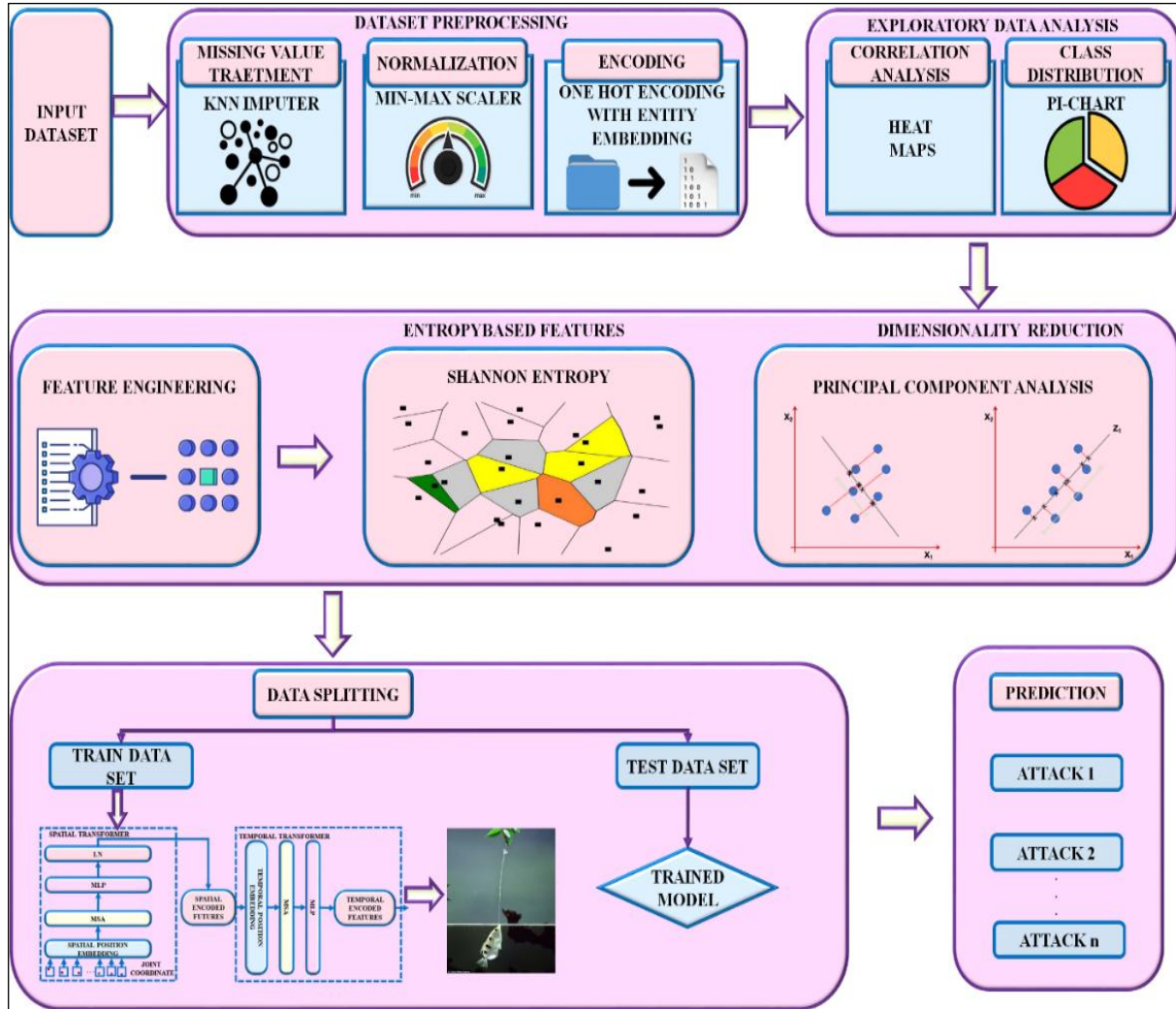


Figure 1: Block diagram of the proposed method.

Source: Authors, (2026).

The first step involves gathering raw network traffic data and performing the full preprocessing workflow. This includes Performing KNN-based missing value imputation, Min-Max Normalization and a Hybrid encoding scheme of one-hot encoding and entity embedding to enable the creation of high quality, meaningful representations. Following the preprocessed data, an EDA conducted to analyse the class imbalance using pie charts and discover the relationship between the Inter-Features by means of visually distinguishing between features created from a graphical heatmap using the correlation coefficient. To enable more expressiveness of the features, Shannon Entropy is used to create additional attributes which express entropy. Additionally, by using PCA, the number of dimensions is reduced, eliminating redundant features and improving computational efficiency for the next stage. The final preprocessed dataset is then split into a train dataset and a test dataset, where the train dataset is used to train the core architecture, called FEA-CasSTFormerNet. The FEA-CasSTFormerNet model utilises FEA, which provides recursive memory aware Attention, and Cascaded Spatio-Temporal Transformer (CasSTFormer), which captures the Deep-Spatial and temporal dependences of the network traffic. Finally, hyperparameter tuning is conducted on the model using Archerfish Hunting Optimization (AfHO) algorithm, which provides a great balance between exploration and exploitation. The system's capacity to correctly categorize various attack types with high accuracy, low false-positive rates, and strong generalization capabilities is demonstrated by experimental findings on benchmark datasets.

### III. PROPOSED SYSTEM MODELLING

#### III.1 DATA PREPROCESSING

##### III.1.1 KNN Imputer

Datasets used for IDS ensure missing values as a result of lost packets or errors that occurred during logging. Reliable detection only occur if these missing values are handled properly. To handle missing values KNN Imputer is used, which estimates missing data by observing the k-nearest neighbor-based on the weighted Euclidean distance between the present features coordinates. The distance calculation is performed using the provided equation,

$$D_{xy} = \sqrt{\text{weight} \times \text{squared distance from present coordinates}} \quad (1)$$

$$\text{Where, } \text{weight} = \frac{\text{total number of coordinates}}{\text{number of present coordinates}} \quad (2)$$

The KNN Imputer uses this weighted distance to accurately reconstruct partial network traffic data, maintaining essential patterns and increasing the performance of the IDS.

##### III.1.2 Min-Max Scaler

Network traffic features is scaled extremely different, therefore there must be a method for normalizing different scales. The types of variations in the traffic feature hinder the ability of a DL model to develop a strong consideration of the features used during its learning process. The Min-Max scaler normalization method works well on intrusion detection data because it normalizes all feature values to a common scale within an established range between 0 and 1. This is done to avoid data that contains larger numbers from dominating the learning process of a DL algorithm, thus permitting all features equal weight in determining the output of the network-based detection system. Normalized values generated by the Min-Max Scaler are based on the following mathematical equations,

$$X_{std} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \quad (3)$$

$$X_{scaled} = X_{std} \times (X_{max} - X_{min}) + X_{min} \quad (4)$$

Here,  $X_{min}$  and  $X_{max}$  represent the minimum and maximum values of the traffic feature being considered, respectively. It is important to note that the application of the Min-Max scaler transformation across the entire data set create all of the traffic features values to have the same scaled value, thus helping the efficiency of the model and increasing the model's ability to generalize on network traffic.

##### III.1.3 One-Hot Encoding with Entity Embedding

In intelligent NIDS, DL for processing the categorical network traffic features such as the Protocol Type, Service and Flag, these features need to be converted from their categorical format into numerical values. A simple and effective way to accomplish this conversion is through using the one-hot encoding technique, where each of the categories is transformed into a binary vector representation of a specific length representing each category in the original feature. One-hot encoding is accomplished very quickly, does not require a lot of processing and retains all of the information about each category so the data directly passed to the neural networks to be used for training and testing the models. To explain more complex relationships between the categories of a specific feature the technique of entity embedding is used. Entity embeddings are represented mathematically using the following equation,

$$e_i: x_i \rightarrow X_i \quad (5)$$

Where,  $x_i$  is the original categorical value, and  $X_i$  is the embedded vector of that categorical value. From Equation (6), it shows that the mapping method is mathematically equal to multiplying a one-hot encoded vector by a linear layer. This relationship is expressed mathematically as,

$$X_i = \sum_{\alpha} W_{\alpha\beta} \delta_{x_i\alpha} = W_{x_i\beta} \quad (6)$$

Where,  $\alpha, \beta, \delta$  denotes the Kronecker delta that returns 1 when the indices are equal and 0 when not equal, and  $W_{\alpha\beta}$  shows the learnable weights of the embedding. The weights are optimized during the training of a NIDS to enable similar behaviours from the network to be embedded close to one another within the embedding space. As a result, the combination of one-hot encoding and entity embedding allows the system to use an efficient method of representing its input while still allowing the model to learn structural relationships between the categorical features, thus improving its ability to accurately detect intrusions.

#### III.2 EXPLORATORY DATA ANALYSIS

##### III.2.1 Heatmap

Heatmaps are used in NIDS to provide a compact and easily interpreted view of significant amounts of network traffic data. Each row represents a specific network flow, while columns represent specific traffic features or model outputs for those flows, with the color of the cell denoting a value for the traffic feature or a detection score from the model.

The heatmap allows the user to quickly identify and visualize correlations among flows based on their behavioral patterns (both benign and malicious) via hierarchical clustering of the flows. Furthermore, heatmaps contain a lot of information per square inch, and therefore allow the user to visually identify which features are important, which correlations exist among multiple flows, and where conflicting detections occur across all flows within the large dataset. Heatmaps are, therefore, a very useful visualization tool for understanding and improving the performance of intrusion detection systems.

### III.2.2 Pie Charts

Pie charts employed in NIDS, visualising the relative percentage breakdown of the various forms of network traffic and the attack classes being detected. The section of the pie representing each class such as benign, DoS, probing, R2L and U2R is represented in terms of the size of the pie section that is proportionate to the total percentage of flows that match those attack classes. The diagram of pie chart is shown in figure 2.

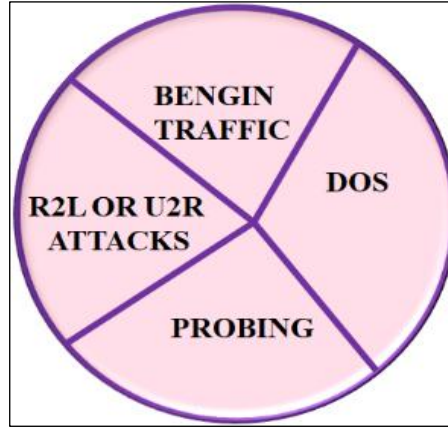


Figure 2: Pie chart.  
Source: Authors, (2026).

The use of this circular representation enables analysts to assess the types of attacks exist within a given data set and to interpret the detections from a particular NIDS are distributed. Pie charts are useful in providing a high level of visualisation about traffic composition and allow for the ability to assess overall detection capabilities, the existence of bias towards certain types of attacks and the impact of threats to the network.

## III.3 FEATURE ENGINEERING

### III.3.1 Shannon Entropy

The Shannon Entropy mathematically relates to the uncertainty associated with a given probability distribution. Entropy, originally used to quantify uncertainty of a collection of probability distributions, in detecting network intrusion attacks using the features of network traffic. Since features of network traffic data collection are interpreted as random variables, the Shannon entropy is applied. Any individual attribute of network traffic, denoted by the variable  $X$ , have an associated entropy which is calculated using the following equation,

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \quad (7)$$

A network's Probability Density Function (PDF) contains the uncertainty of its network traffic features such as IP addresses, Ports, Packet sizes, and Protocols. Normal or predictable network activities show stable distributions and maintain the relative entropy within a predictable or generated range. Disruptive behaviour increases the entropy of the PDF due to the increased variation of values within a distribution, whereas Flooding or DDoS type behaviours repeat patterns which produce low-entropy PDFs. The functional form of the Shannon is mathematically expressed as,

$$H(p_1, \dots, p_n) = -K \sum_{i=1}^n p_i \log_2 p_i \quad (8)$$

Network traffic analysis requires a range of isolates while maintaining a monotonic progression. The entropy calculations used in intelligent intruder detection systems consider only the probability values of data and therefore are both permutation-invariant and robust to label changes. The application of entropy mechanisms for intelligent intrusion detection systems facilitates inexpensive, real-time, and scalable detection of abnormal network behaviour.

### III.3.2 Principal Component Analysis

In identifying intrusions using a neural network based on its identification model from an IDS on a network, traffic data is represented by matrix  $X$  of dimension  $I \times J$ , where  $I$  is the number of network sessions, and  $J$  is the number of features related to the sample such as protocol type, packet size and duration. Each feature vector associated with the samples be combined in a linear combination of the features to create a new variable called principal component defined as  $t = w_1 x_1 + \dots + w_j x_j$ , or symbolically  $t = Xw$ , where  $w$  is the weight vector of the components. PCA maximizes the variance of the principal component  $t$  while ensuring that  $\|w\| = 1$ , which is formally expressed by,

$$\arg \max_{\|w\|=1} \text{var}(t) = \arg \max_{\|w\|=1} w^T X^T X w \quad (9)$$

The constraint  $\|w\| = 1$  prevents solutions where  $w$  have any positive value. The solution is the eigenvector of  $X^T X$  associated with the highest eigenvalue, called the first principal component. This first principal component represents the dominating mode of original, normal network behavior. Any newly collected sample traffic mapped to the location in a vector space defined by the first principal component. Any traffic point that lies far from the normal-pattern variance of the principal component is identified as an abnormal traffic point or an attack vector. PCA provides to capture all of the variance of original normal network behavior and to transform that captured data set to a lower-dimensional representation while maintaining the underlying structure of network activity, allowing for the identification of anomalous or attack-type behavior.

### III.4 DATA SPLITTING

#### III.4.1 FEA-CasSTFormerNet

The proposed FEA-CasSTFormerNet approach to modelling network traffic uses an extraction of features as marks in a similar manner as specifying joints in transformer based position models. The model treats a network feature vector  $x_t^j \in R^F$ , as an individual mark. First, the traffic feature vector must be projected into a higher dimensional embedding through a learnable projection matrix in order for the model to learn better representations of the traffic behaviour that is observed.

$$Z_0^t = [x_1 E, x_2 E, \dots, x_j E] + E_{SPos} \quad (10)$$

Where,  $E$  is the projection matrix and  $E_{SPos}$  ensures that the semantic identity of each traffic feature is preserved. The embeddings produced from  $E$  are then passed through multiple cascaded spatial attention layers,

$$Z_l^t = \text{MSA}(\text{LN}(Z_{l-1}^t)) + Z_{l-1}^t \quad (11)$$

$$Z_l^t = \text{MLP}(\text{LN}(Z_l^t)) + Z_l^t \quad (12)$$

Which learn complex intra time step relations such as the correlations between packet sizes, protocol flags, timing patterns and variations in entropy. The spatial outputs are flattened into vectors  $Z^t \in R^{1 \times (J \cdot C)}$ , and concatenated together across time to form,

$$Z_0 = \{Z^1, Z^2, \dots, Z^T\} \quad (13)$$

Which, have temporal positional embeddings added to them before entering the temporal transformer. Diagram of FEA-CasSTFormerNet is shown in figure3.

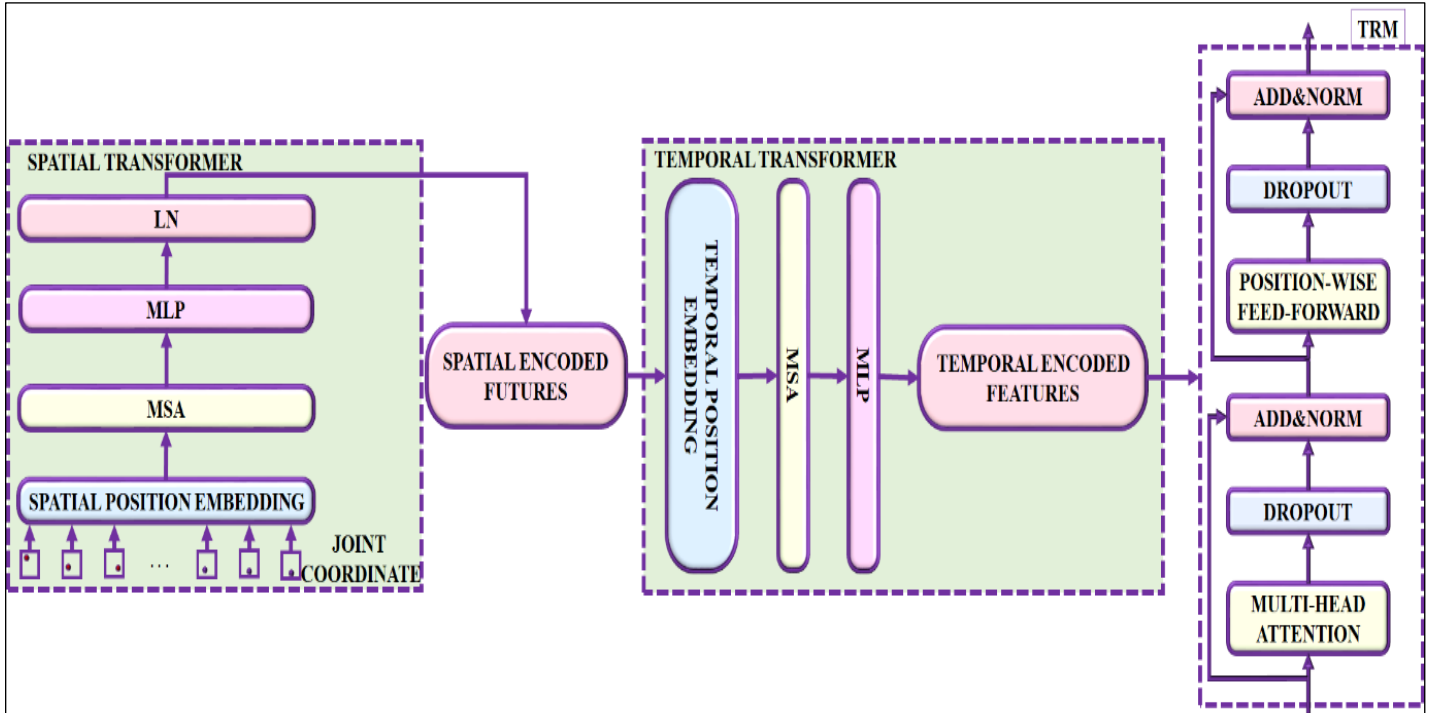


Figure 3: Diagram of FEA-CasSTFormerNet.  
Source: Authors, (2026).

At this stage, multi-head temporal attention is used to capture long-term temporal relationships and changing behavioural signatures within the traffic sequence, producing a single globally encoded temporal representation  $Y \in R^{T \times (J.C)}$ . By integrating fractal dimension-based measures of complexity with improved feature representations and cascaded spatial-temporal attention mechanisms, FEA-CasSTFormerNet produces very distinct feature embeddings that detect complex, multi-stage, and unusual patterns of intrusion that typical intrusion detection systems fail to detect.

#### III.4.2 Archerfish Hunting Algorithm

This research uses the AfHO algorithm to identify the best combination of parameters to use in an intelligent IDS. The AfHO algorithm replicates the hunting approach of the archer fish, which use two main methods to hunt for food either shooting water at large insects (exploratory behaviour) or jumping into water to catch smaller insects (exploitative behaviour). Regarding the IDS, each archer fish is an individual candidate configuration for the IDS, while the corresponding insect is considered a good-performing configuration that provides the best possible detection accuracy.

Pseudocode: AfHO

```

Start
// 1. Parameters
Initialize population_size, max_iterations
Initialize lower_bound, upper_bound
Initialize best_solution = NULL, best_fitness = -Infinity
// 2. Initialize population
For each fish i in population_size
Initialize position[i] randomly within bounds
Evaluate fitness[i]
If fitness[i] > best_fitness
best_fitness = fitness[i]
best_solution = position[i]
EndIf
EndFor
// 3. Main loop
For iter = 1 to max_iterations
For each fish i in population_size
Generate random number r in [0,1]
// 4. Exploration (hunting new area)
If r < 0.5
new_position = position[i] + random_step()
Else
// 5. Exploitation (approach prey)
new_position = position[i] + rand() * (best_solution - position[i])
EndIf
// 6. Boundary check
If new_position < lower_bound
new_position = lower_bound
ElseIf new_position > upper_bound
new_position = upper_bound
EndIf
// 7. Fitness evaluation
new_fitness = EvaluateFitness(new_position)
// 8. Update position
If new_fitness > fitness[i]
position[i] = new_position
fitness[i] = new_fitness
EndIf
// 9. Update global best
If new_fitness > best_fitness
best_fitness = new_fitness
best_solution = new_position
EndIf
EndFor
EndFor
Return best_solution, best_fitness
End

```

Source: Authors, (2026).

The step-by-step procedure for the algorithm begins by constructing an initial population then, the algorithm generates a respective position for all fish based on the identification of the  $i^{th}$  fish and its alignment with the target insect.

$$P(i, 0) = [P_1^L + \sigma_1(P_1^U - P_1^L), \dots, P_d^L + \sigma_d(P_d^U - P_d^L)] \quad (14)$$

Table 2: Parameters of AfHO

| PARAMETER                                  | VALUE                 |
|--------------------------------------------|-----------------------|
| Population Size (N)                        | 30                    |
| Maximum Iterations (Max_Iter)              | 50                    |
| Lower Bound (LB)                           | -10                   |
| Upper Bound (UB)                           | 10                    |
| Exploration Probability (p)                | 0.5                   |
| STEP SIZE (Step)                           | $0.1 \times (UB-LB)$  |
| Learning Rate/ Attack Factor ( $\alpha$ )  | 0.001                 |
| Random Number (R)                          | $0 \leq r \leq 1$     |
| Inertia Weight/ Momentum Factor (W)        | 0.5                   |
| Scaling Factor For Exploration ( $\beta$ ) | 0.1                   |
| Convergence Threshold ( $\varepsilon$ )    | $1e^{-6}$             |
| Probability Of Global Search (Pg)          | 0.3                   |
| Maximum Step Adjustment (Step_Max)         | $0.2 \times (UB-LB)$  |
| Minimum Step Adjustment(Step_Min)          | $0.01 \times (UB-LB)$ |
| Batch Size                                 | 32                    |
| Epoch                                      | 10                    |

Source: Authors, (2026).

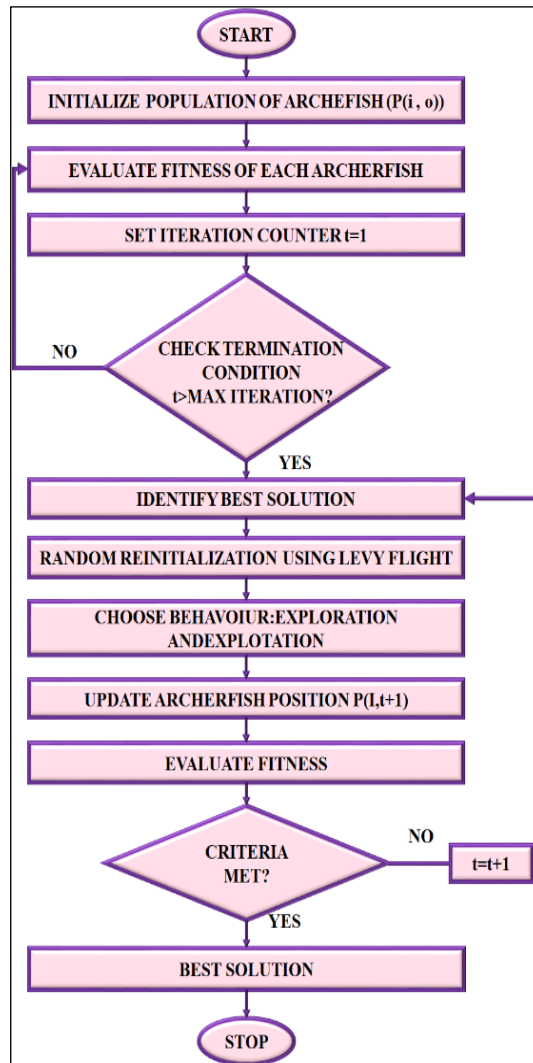


Figure 4: Flowchart of AfHO algorithm.

Source: Authors, (2026).

In exploration, the archerfish position is updated based on the results of shooting behaviour, allowing the search agents to explore potentially more optimal areas of the IDS parameters, while still maintaining an adequate amount of randomness to prevent premature convergence to a local optimum.

$$P(i, t + 1) = -(P(i, t) - P_{in}(i, t))e^{-\|P_{in}(j, t) - p(i, t)\|^2} + P(i, t) \quad (15)$$

In exploitation, the jumping behaviour fine-tunes the potential IDS configurations, concentrating the search algorithm on the most likely optimal solutions, and by continuing the localized refinement of those solutions to increase their accuracy.

$$P(i, t + 1) = -(P(i, t) - P_{in}(i, t))e^{-\|P_{in}(j, t) - p(i, t)\|^2} + L(i, t) \quad (16)$$

If a position is stagnant, an alternate set of IDS configurations is created, ensuring that the search continues to explore additional parameter space rather than becoming stuck at a local prime.

$$P(i, t + 1) = P(i, t) + \sigma \left[ P_1 v_1^{\frac{1}{\beta}}, \dots, P_d v_d^{\frac{1}{\beta}} \right] \quad (17)$$

Where,  $P_i \sim fn(0, \gamma^2)$ ,  $\beta = 1.5$ . As a result of this balance between the two types of behaviour, AfHO exhaustively searches the parameter space and finds the best-performing model for intrusion detection, providing a more accurate balance of the amount of global exploration versus localized refinement.

#### IV. RESULT AND DISCUSSION

The propose system uses a KDD99 Intrusion Detection dataset to evaluate the proposed FEA-CasSTFormerNet model. Because of its wide usage among researchers studying network security, it is often considered a benchmark dataset for evaluating performance. The dataset consists of a total of 494,020 network traffic samples, from which 42 separate features extracted with each record assigned an attack or normal label. The features in the KDD99 Intrusion Detection dataset fall into several categories including; Basic Features, Content Features, Traffic Features, Host Statistics. For training and testing the FEA-CasSTFormerNet model, the dataset is split into 80% training data and 20% testing data, providing a required amount of data for performance assessment when detecting intrusions.

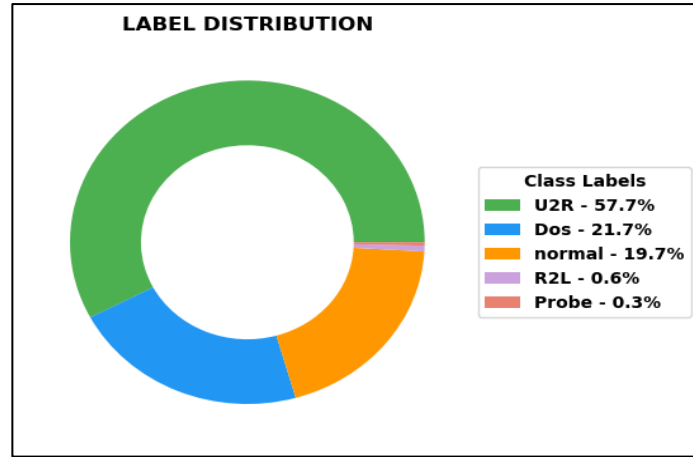


Figure 5: Label distribution.  
Source: Authors, (2026).

Figure 5 shows the label distribution. The distribution of the labels varies significantly. It is highly skewed toward U2R (57.7%) as the primary class, with DoS (21.7%) second, and normal traffic (19.7%) coming in third place while R2L (0.6%) and Probe (0.3%) are extremely limited.

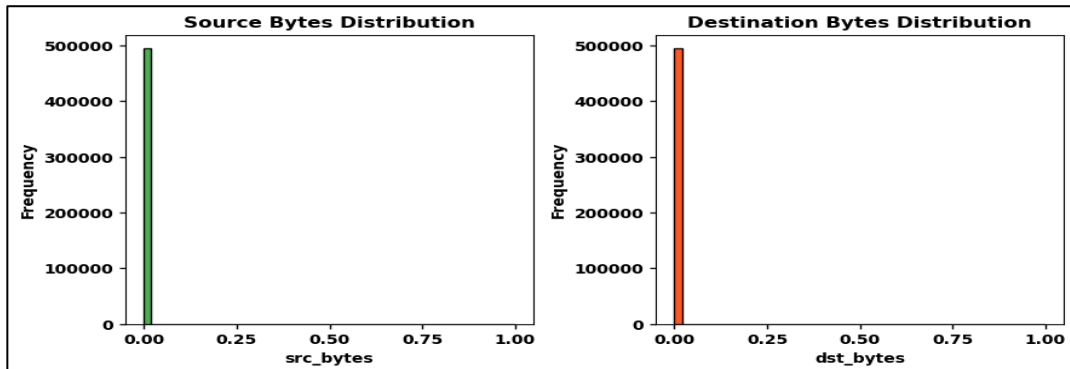


Figure 6: Source and destination bytes distribution.  
Source: Authors, (2026).

Figure 6 depicts the source and destination bytes distribution. The distributions of the source bytes and destination bytes are also highly skewed toward zero, indicating that the majority of network connections have low data transferred both ways.

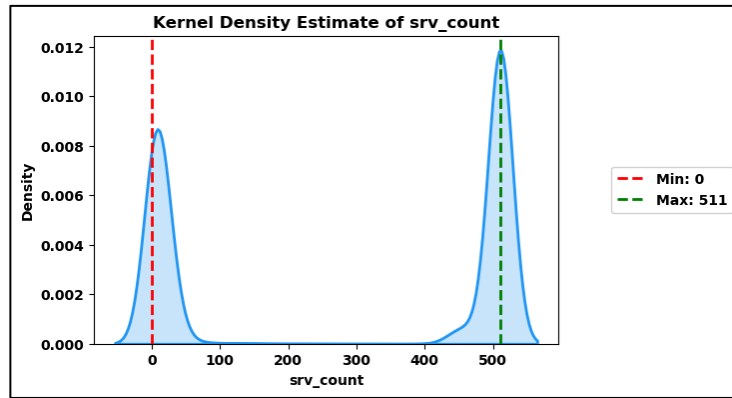


Figure 7: Kernel density estimation.  
Source: Authors, (2026).

Figure 7 illustrates a Kernel Density Estimate (KDE) plot has been created to visualize the `srv_count` variable. From this plot, it is apparent that a bimodal distribution is present; with one peak around zero, and a second peak around 511. The dashed vertical lines indicate where the data clusters heavily around these two extreme values (0 - red, and 511 - green).

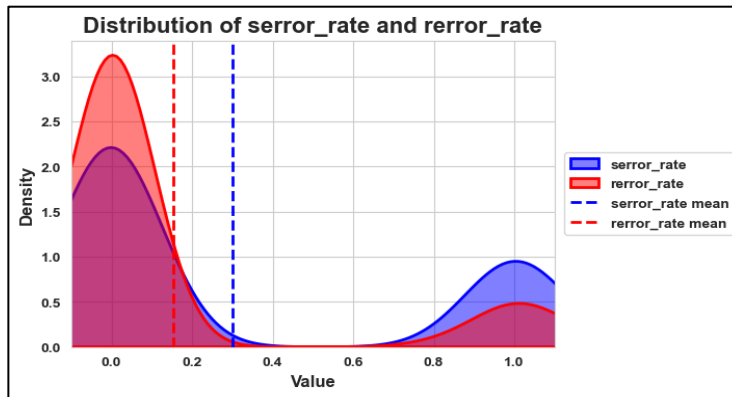


Figure 8: Distribution of classes.  
Source: Authors, (2026).

This figure 8 shows the Kernel Density Estimate (KDE) plots of both the error rate (blue) and error rate (red). Both distributions are highly concentrated in the area close to 0. However, the error rate has a very peak and the error rate has a secondary rise up to 1.0. The dashed vertical lines represent the mean of both distributions. The error rate mean value is lower than the error rate mean value.

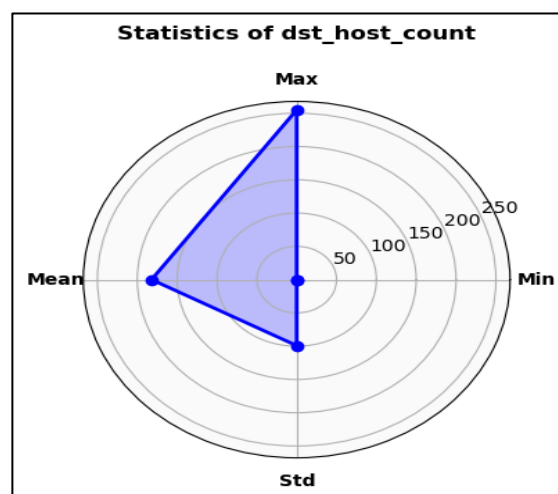


Figure 9: Statistics of `dst_host_count`.  
Source: Authors, (2026).

This figure 9 provides an overview of the `dst_host_count`. The `dst_host_count` shows a low minimum value near 0, moderate standard deviation, average near the middle, and high maximum value, indicating that there is a significant range and that there is a lot of variation for the `dst_host_count` feature.

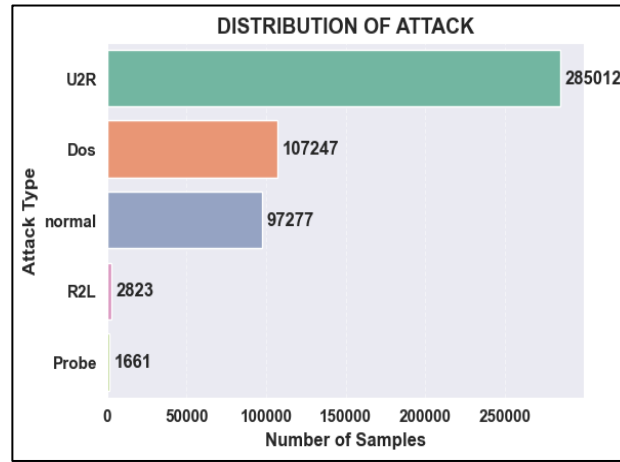


Figure 10: Distribution of attack.

Source: Authors, (2026).

This figure 10 demonstrates the number of samples for all network attack types in the dataset. U2R attacks represent the majority of the sample size, followed by DoS and normal traffic; R2L and Probe attacks are present in many fewer samples than U2Rs, indicating that the dataset is unbalanced.

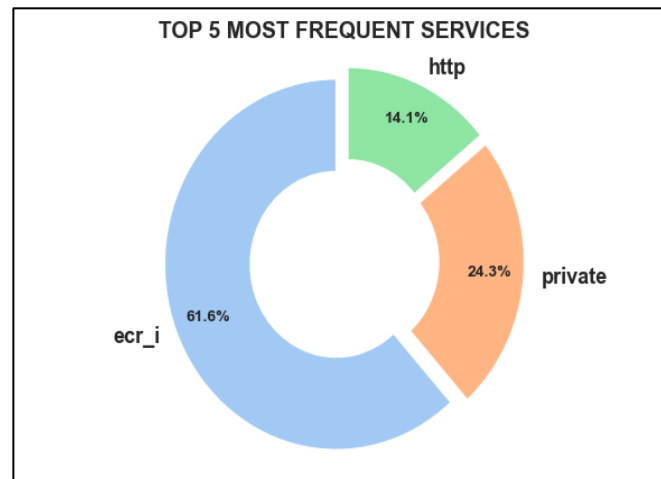


Figure 11: Diagram of top 5 most frequent services.

Source: Authors, (2026).

This figure 11 displays the number of occurrences of the top 5 network services. The ecr\_i service represents 61.6% of the occurrences, followed by the private service with 24.3% and then http service with 14.1%. Therefore, a limited number of services make up the majority of the network traffic.

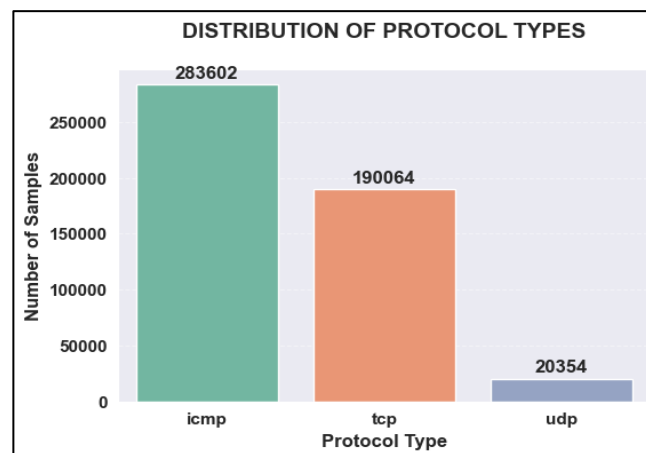


Figure 12: Distribution of protocol types.

Source: Authors, (2026).

Figure 12 illustrating the types of protocols primarily indicates that ICMP is the most frequently recorded, with the highest count (283,602) followed by TCP (190,064) and lastly, UDP (20,354), thus showing a considerable difference in frequency as shown in the results.

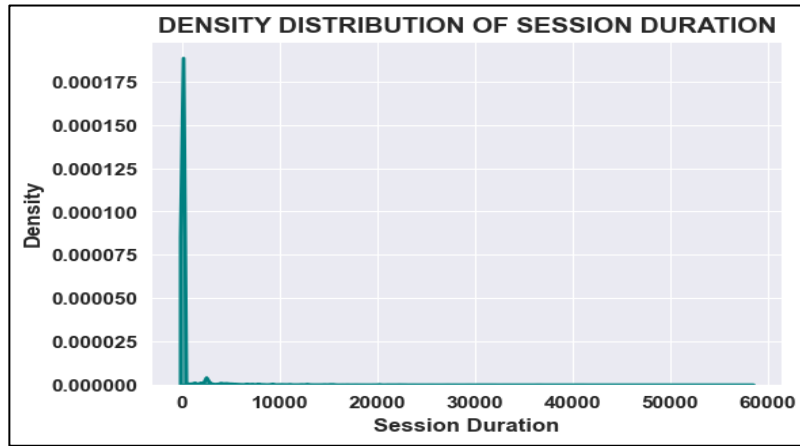


Figure 13: Density distribution of session duration.  
Source: Authors, (2026).

The figure 13 explains that the duration of network sessions has a positive curvature and is primarily concentrated in the short-duration range; that is to say, Sessions that are less than five seconds long create the longest peak but significantly fewer sessions have a session duration over five seconds.

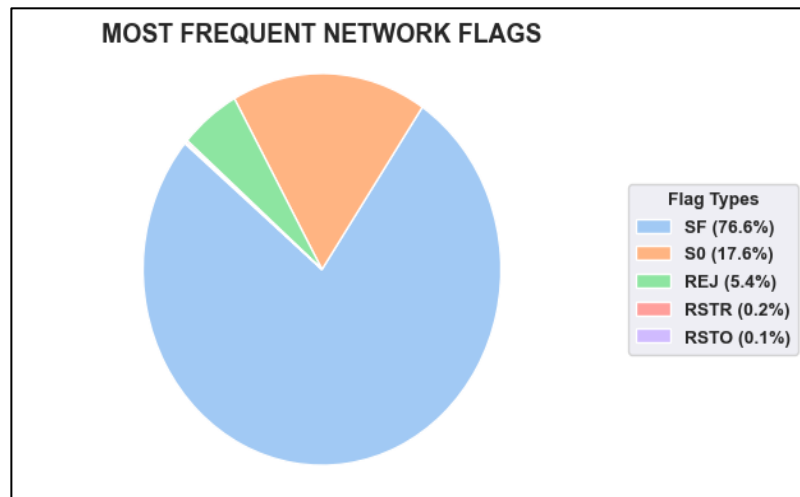


Figure 14: Most frequent network flags.  
Source: Authors, (2026).

The figure 14 establishes that SF flags account for the majority, with the total component resting around 76.6%. The next most prevalent flags are SO (17.6%) and REJ (5.4%). On the contrary, the RSTR flag and RSTO flag are far less prominent, at 0.2% and 0.1%, respectively.

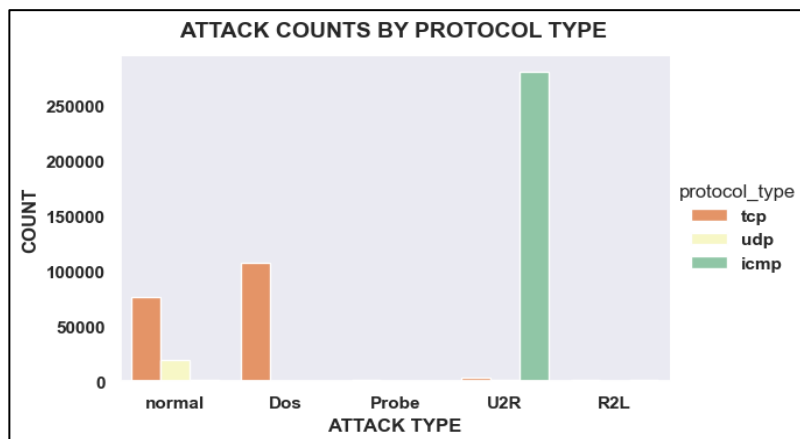


Figure 15: Attack counts by protocol type.  
Source: Authors, (2026).

The figure 15 depicts that TCP, in fact, dominates almost all classifications, so to speak, as well as DoS and Probe attacks. However, ICMP attack types have been found to be primarily linked with U2R attacks, whereas there are comparatively fewer UDP-generated attacks across all attack types.

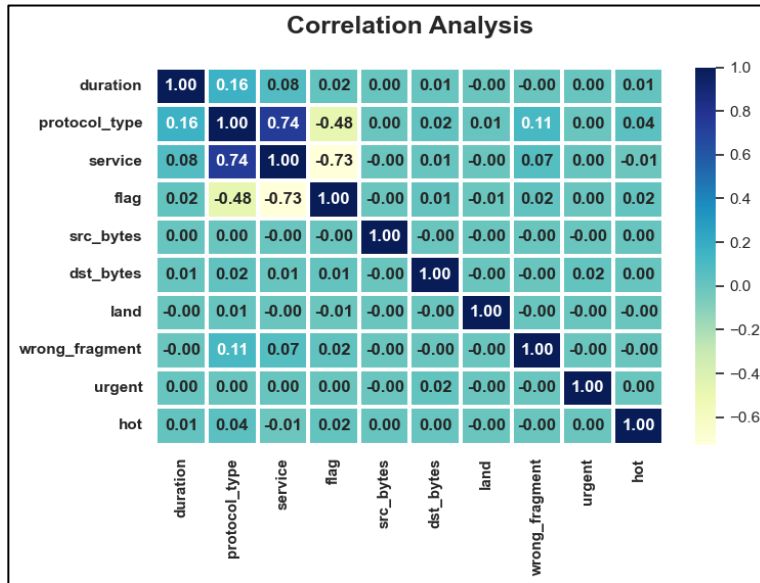


Figure 16: Correlation analysis.  
Source: Authors, (2026).

The figure 16 illustrates that correlation does not necessarily prove causation between several of the various network features. It can be seen that there are pairs of features with very strong correlations (such as protocol type and service, as well as service and flag) but that the majority of the other features do not exhibit any significant linear correlation.

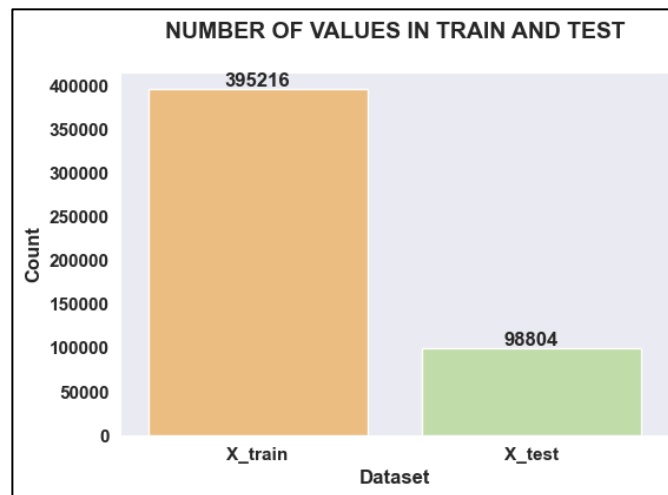


Figure 17: Number of values in train and test.  
Source: Authors, (2026).

The figure 17 compares the dataset sizes for X\_train (395,000 samples) and X\_test (99,000 samples). This is an example of a standard train-test data split.

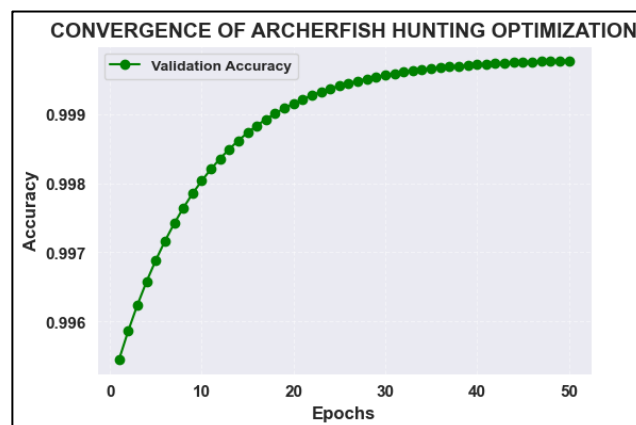


Figure 18: Convergence of Archerfish hunting optimization.  
Source: Authors, (2026).

In figure 18, the validation accuracy continues to increase through the course of 50 epochs. This indicates that the AFHO-FEA-CasSTFormerNet model, which uses the AHO algorithm for optimization, does not plateau during training until it reaches a high level of accuracy.

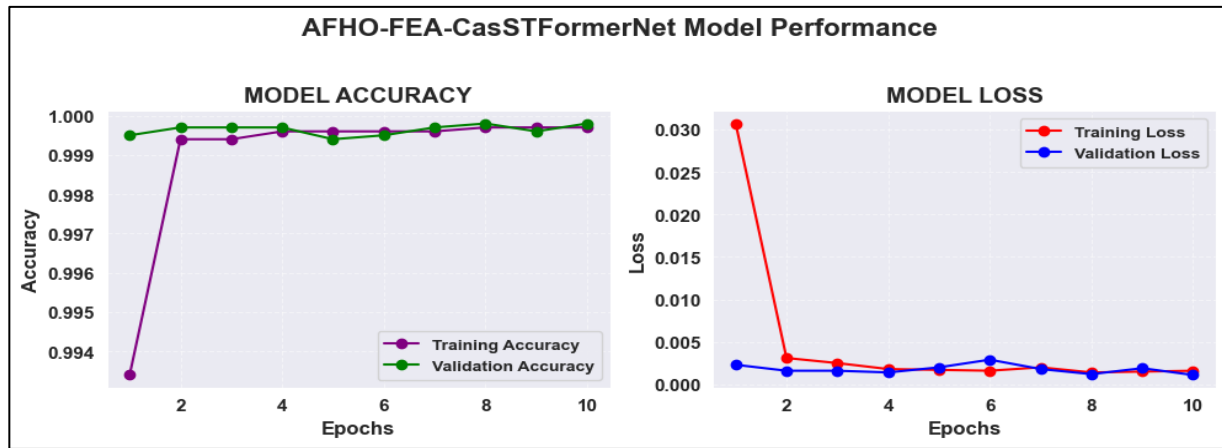


Figure 19: AFHO-FEA-CasSTFormerNet model.  
Source: Authors, (2026).

The loss values in figure 19 dropped significantly during the early training epochs and remained low after that time point. Therefore we know that this model trained well on both sets of data and exhibited little evidence of overfitting.

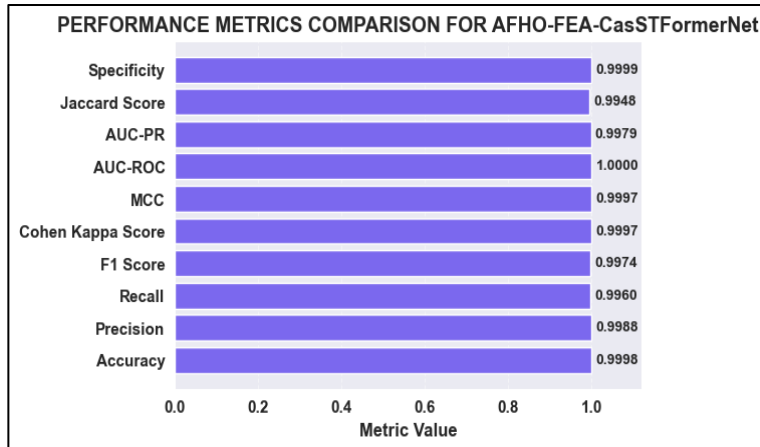


Figure 20: Performance metrics comparison for AFHO-FEA-CasSTFormerNet.  
Source: Authors, (2026).

The figure 20 shows the high performance of the AFHO-FEA-CasSTFormerNet model using seven different evaluation metrics. All evaluations resulted in scores close to perfect (0.996-1.000). The scores for accuracy, precision, recall, F1-score, Matthews Correlation Coefficient (MCC), area under the receiver operating characteristic curve (AUC-ROC), and specificity suggest that the model performed well to well-balanced classify within the context of detecting various types of attacks.

CONFUSION MATRIX FOR AFHO-FEA-CasSTFormerNet

| True Label | Normal | DoS             | Probe | R2L   | U2R   |     |
|------------|--------|-----------------|-------|-------|-------|-----|
|            | 21447  | 0               | 0     | 1     | 1     |     |
|            | 1      | 329             | 0     | 0     | 2     |     |
|            | 0      | 0               | 559   | 0     | 6     |     |
|            | 0      | 0               | 0     | 57002 | 0     |     |
|            | 2      | 0               | 3     | 3     | 19448 |     |
|            |        | Predicted Label |       |       |       |     |
|            |        | Normal          | DoS   | Probe | R2L   | U2R |

Figure 21: Confusion matrix for AFHO-FEA-CasSTFormerNet.  
Source: Authors, (2026).

The figure 21 represents accurate AFHO-FEA-CasSTFormerNet model classified samples into five different categories (Normal, DoS, Probe, R2L, U2R). Almost all samples are classified correctly because very few exist that have been misclassified (about 0 to 3, per class). Hence, the AFHO-FEA-CasSTFormerNet model is capable of identifying both various types of attacks and normal traffic with great accuracy and is unlikely to misclassify them, given the high degree of accuracy achieved.

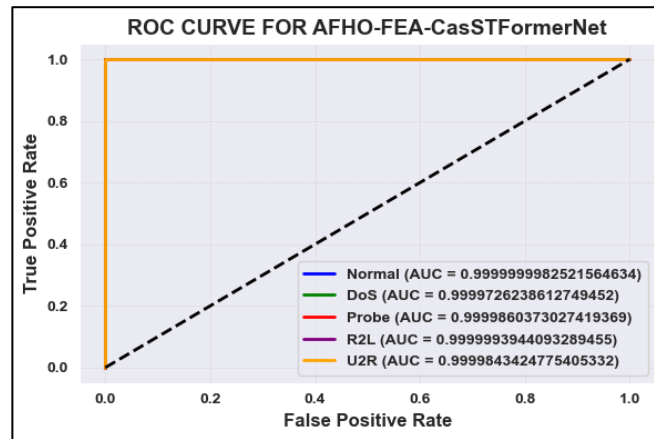


Figure 22: ROC curve for AFHO-FEA-CasSTFormerNet.

Source: Authors, (2026).

The ROC curve relates to the performance of the AFHO-FEA-CasSTFormerNet system in figure 22. Almost all of the five classes of traffic (Normal, DoS, Probe, R2L, U2R) are located in the upper-left corner of the ROC curve, indicating very good classification capabilities for each class. The AUC (Area Under the Curve) for each class is very close to 1.0, indicating excellent ability to discriminate between different classes.

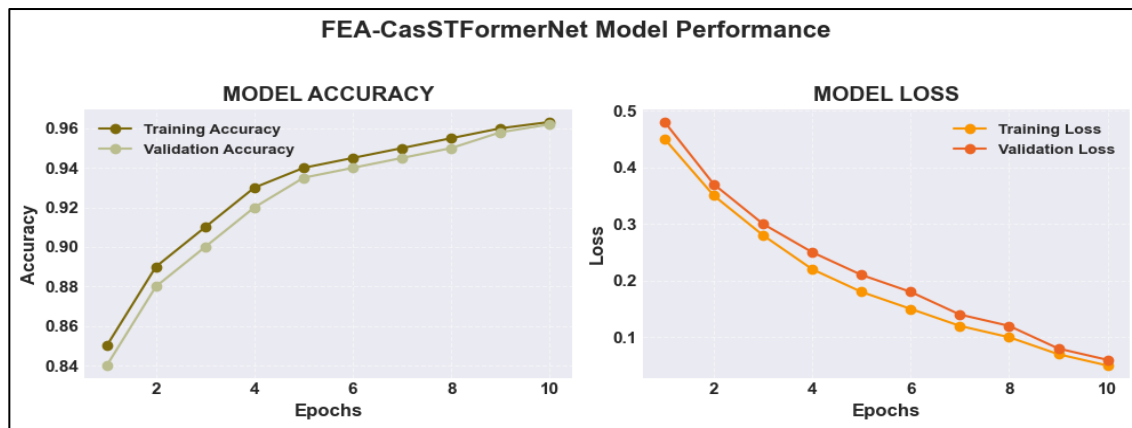


Figure 23: FEA-CasSTFormerNet model performance.

Source: Authors, (2026).

Figure 23 depicts the FEA-CasSTFormerNet model performance. The training/validation results for the FEA-CasSTFormerNet training set show that accuracy was steadily improving during the 10 epochs, while both training and validation losses were consistently decreasing. The loss of the validation set closely follows that of the training set, which demonstrates that the algorithm is learning the training set without significant overfitting.

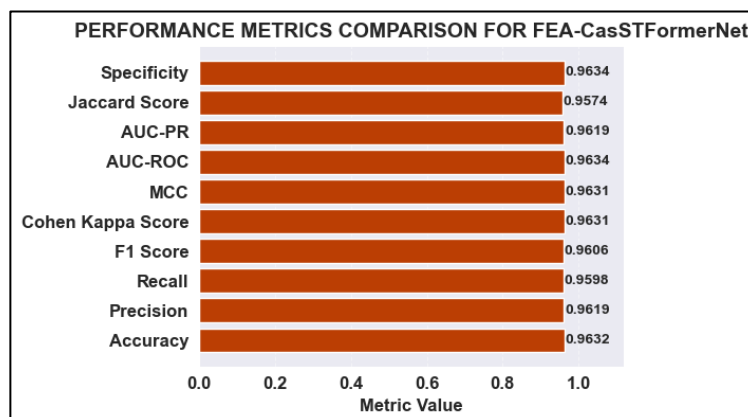


Figure 24: Performance metrics comparison for FEA-CasSTFormerNet.

Source: Authors, (2026).

The performance metrics for the FEA-CasSTFormerNet model shown in figure 24 are very high overall, and range from approximately 0.96 for accuracy, precision, and recall performance, to moderately balanced across all classes.

| CONFUSION MATRIX FOR FEA-CasSTFormerNet |        |     |       |       |       |
|-----------------------------------------|--------|-----|-------|-------|-------|
| True Label                              | Normal | DoS | Probe | R2L   | U2R   |
| Normal                                  | 21000  | 200 | 120   | 80    | 49    |
| DoS                                     | 20     | 310 | 10    | 15    | 25    |
| Probe                                   | 12     | 10  | 525   | 8     | 10    |
| R2L                                     | 40     | 22  | 15    | 54600 | 325   |
| U2R                                     | 30     | 15  | 20    | 40    | 19200 |
| Predicted Label                         |        |     |       |       |       |

Figure 25: Confusion matrix for FEA-CasSTFormerNet.  
Source: Authors, (2026).

In figure 25 the confusion matrix shown clearly that most sample classifications were accurate, and that the Normal, R2L, and U2R classes received a significant proportion of accurate classifications, while the attack classifications received only small proportions of misclassifications.

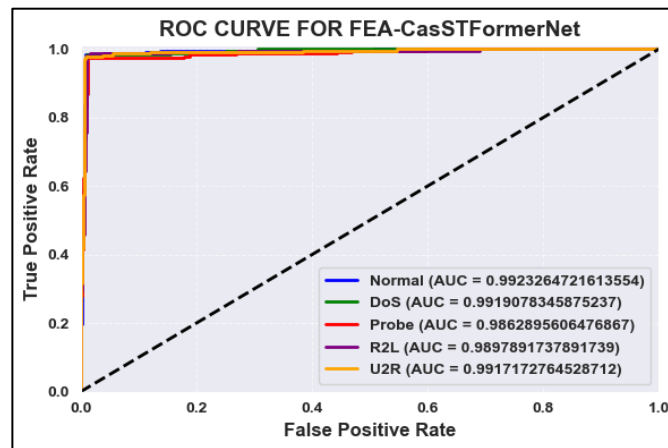


Figure 26: ROC curve for FEA-CasSTFormerNet.  
Source: Authors, (2026).

In figure 26 the ROC curve shows that FEA-CasSTFormerNet provides excellent classification performance for all traffic classes with an AUC value of greater than 0.95 for each class. This indicates very high true positive rates and low false positive rates and demonstrates the ability of the model to accurately classify Normal, DoS, Probe, R2L, and U2R traffic.

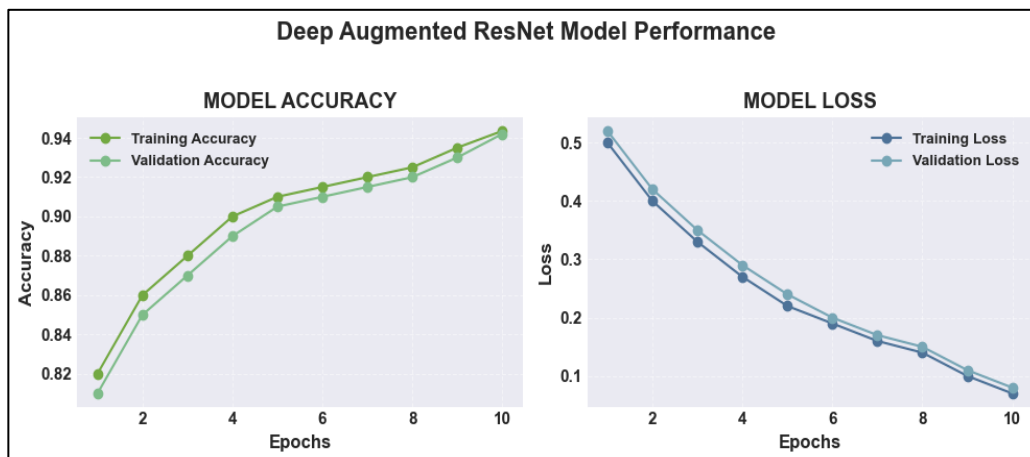


Figure 27: Deep augmented ResNet model performance.  
Source: Authors, (2026).

Figure 27 displays the Deep augmented ResNet model performance. The statistical analysis revealed that the Deep Augmented ResNet exhibited consistent improvement throughout both the Training/Validation phases, with steadily increasing accuracies and decreasing losses for both Train and Validation data sets suggesting that the model was learning at a steady and effective rate.

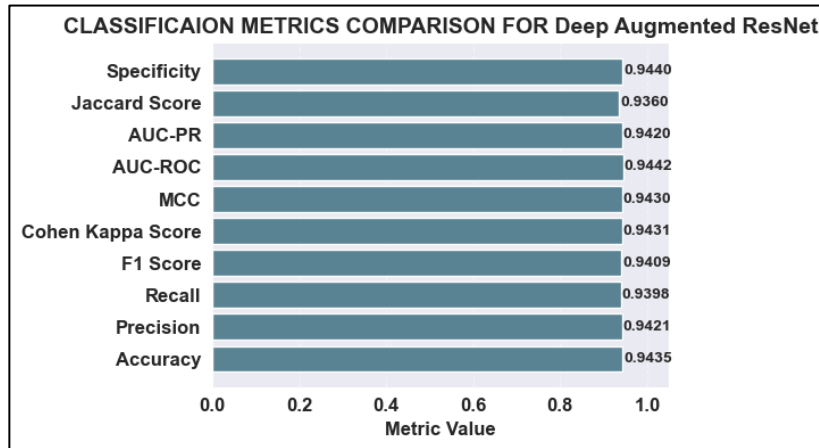


Figure 28: Classification metrics comparison for Deep augmented ResNet.  
Source: Authors, (2026).

The figure 28 displayed the graphical representation indicate that the Deep Augmented ResNet is a highly accurate, reliable and balanced classifier with nearly all of the metrics scoring above 0.93.

| True Label | Normal | DoS             | Probe | R2L   | U2R   |  |
|------------|--------|-----------------|-------|-------|-------|--|
|            | 20500  | 350             | 150   | 100   | 80    |  |
|            | 35     | 295             | 22    | 18    | 20    |  |
|            | 20     | 18              | 505   | 15    | 17    |  |
|            | 90     | 60              | 40    | 54000 | 820   |  |
|            | 55     | 35              | 30    | 65    | 18950 |  |
|            |        | Predicted Label |       |       |       |  |
|            | Normal | DoS             | Probe | R2L   | U2R   |  |

Figure 29: Confusion matrix for Deep augmented ResNet.  
Source: Authors, (2026).

The figure 29 indicates that the Deep Augmented ResNet effectively classifies nearly all samples correctly across each of the five categories, particularly with regards to its very high number of true positives in the R2L, U2R and Normal classification categories. The number of misclassified samples is also extremely low, demonstrating that the Deep Augmented ResNet is capable of accurately distinguishing between diverse forms of attacks and normal network traffic, with only minimal confusion between closely related classes.

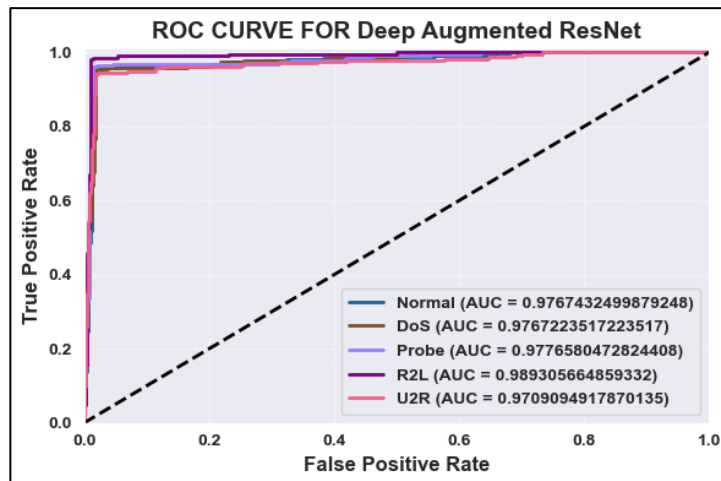


Figure 30: ROC curve for Deep augmented ResNet.  
Source: Authors, (2026).

In figure 30 the visual representation of the ROC curves shows an excellent performance by the Deep Augmented ResNet in terms of each of the five categories, with the AUC values varying between approximately 0.97-0.99. All of the ROC curves are located in proximity to the upper left corner of the graph, indicating a very high True Positive Rate and a very low False Positive Rate for all classes, thus demonstrating that the Deep Augmented ResNet is a highly accurate, reliable and balanced classifier across the board.

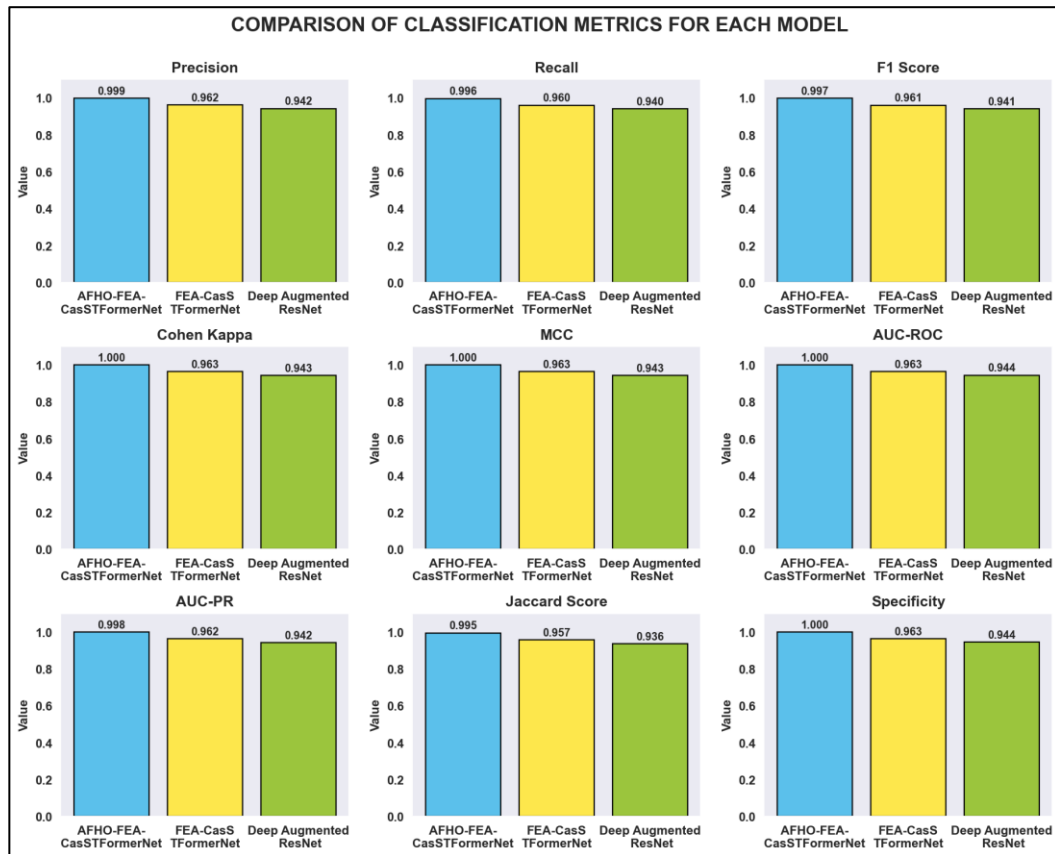


Figure 31: Comparison of classification metrics for each model.

Source: Authors, (2026).

In figure 31 the graphic illustration of the classification metrics indicates that the AFHO-FEA-CasSTFormerNet outperformed the other two models across each of the classification metrics overall, with the FEA-CasSTFormerNet being the second best performing model overall and the Deep Augmented ResNet slightly lower performing model but still being very strong.

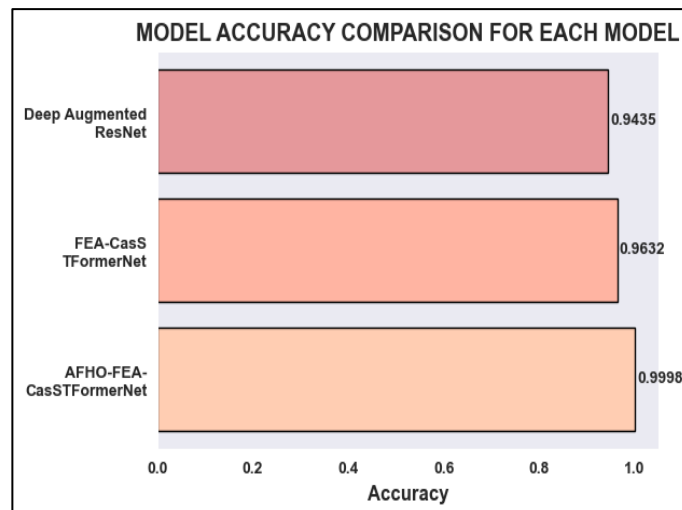


Figure 32: Model accuracy comparison for each model.

Source: Authors, (2026).

Figure 32 compares the model of AFHO-FEA-CasSTFormerNet, FEA-CasSTFormerNet and Deep Augmented ResNet with respective accuracies. AFHO-FEA-CasSTFormerNet (AUC=0.9998) exhibited the best overall performance, while the other two Models exhibited lower accuracy (0.9632 for FEA-CasSTFormerNet; 0.9435 for Deep Augmented ResNet) leading to the inference that the enhanced AFHO Model has a higher level of capability.

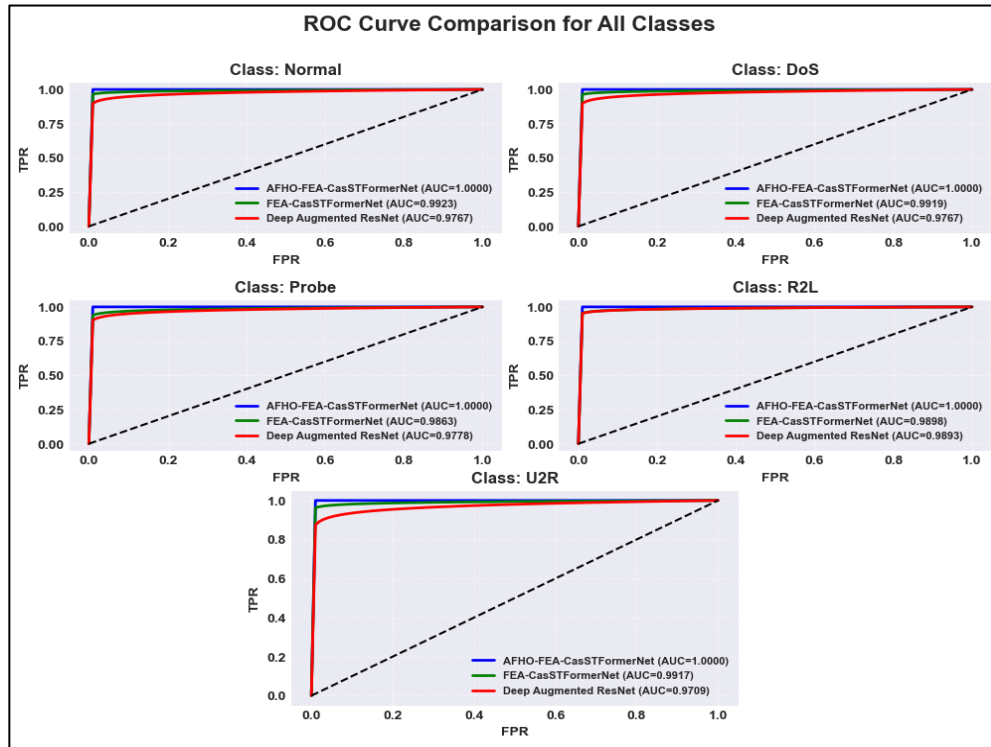


Figure 33: ROC curve comparison for all classes.  
Source: Authors, (2026).

The figure 33 illustrates that AFHO-FEA-CasSTFormerNet performs consistently better than any of the other Models when analyzed according to Class. FEA-CasSTFormerNet exhibited a lesser degree of Class-Level performance, while Deep Augmented ResNet Exhibited the lowest performance levels, in comparison. Therefore, it is reasonable to state that the Enhanced AFHO Model provides the greatest Class-Level Discriminatory capability.

Table 3: Comparative Analysis of Accuracy for Various Methods.

| METHODS       | ACCURACY |
|---------------|----------|
| RNN [21]      | 0.95     |
| TCN [22]      | 0.90     |
| PSO-LSTM [24] | 0.91     |
| GRU [25]      | 0.998    |
| Proposed      | 0.999    |

Source: Authors, (2026).

Table III compares the accuracy for various methods. The suggested approach outperformed RNN (0.95), TCN (0.90), PSO-LSTM (0.91), and GRU (0.998) in network intrusion detection, with the greatest accuracy of 0.999.

Table 4: Performance Metrics of the Proposed System.

| CLASSIFICATION METRICS          | PROPOSED SYSTEM |
|---------------------------------|-----------------|
| ACCURACY                        | 0.9998          |
| PRECISION                       | 0.9988          |
| RECALL                          | 0.9960          |
| F1-SCORE                        | 0.9974          |
| COHEN KAPPA SCORE               | 0.9997          |
| MATHEWS CORRELATION COEFFICIENT | 0.9997          |
| AUC-ROC                         | 1.0000          |
| AUC-PR                          | 0.9979          |
| JACCARD SCORE                   | 0.9948          |
| SPECIFICITY                     | 0.9999          |

Source: Authors, (2026).

In table 4, the proposed system has shown superior classification performance with a very high level of accuracy (99.98%), precision (99.88%), Recall (99.60%), F1-score (99.74%) providing balanced and reliable detection. The level of agreement between the predictions and true labels is almost perfect as demonstrated by the very high Cohen's Kappa (0.9997) and MCC (0.9997). Additionally, the AUC-ROC is perfect (1.0000) and the AUC-PR is exceptionally high (0.9979). Finally, the Jaccard Score (0.9948) and specificity (99.99%) indicate that normal and intrusive traffic is accurately identified.

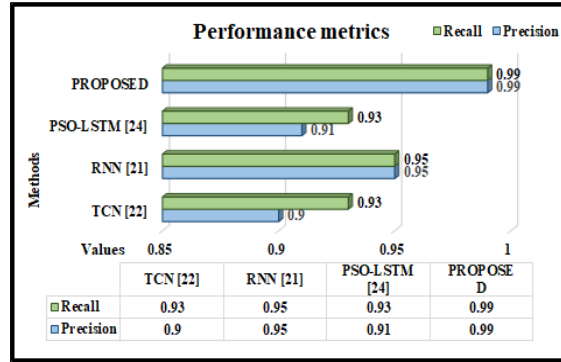


Figure 34: Performance metrics.

Source: Authors, (2026).

According to figure 34, the proposed method significantly outperformed any existing models, achieving the highest precision and recall (0.99 each). The RNN had moderate performance with 0.95; however, the TCN and PSO-LSTM performed worse than RNN, primarily in precision. Overall, the proposed system provides the most reliable and consistent accuracy for detection when compared to other methods.

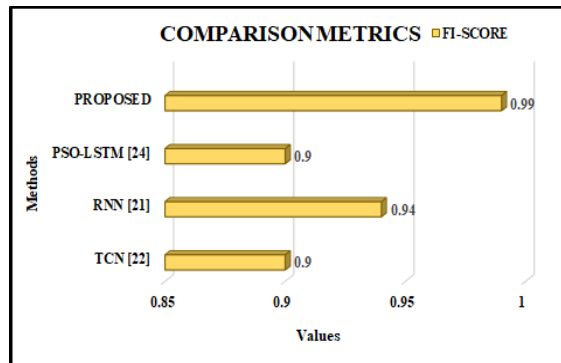


Figure 35: Comparison metrics.

Source: Authors, (2026).

In figure 35, the proposed method demonstrates the highest degree of balance in precision and recall as indicated by the F1-score of 0.99. RNN provides moderate performance scoring 0.94, and PSO-LSTM and TCN provide lower performance than the other methods at scores of 0.90; therefore, the method proposed in this study exhibits superior reliable performance than any other methods described in this paper.

## V. CONCLUSIONS

This system proposes a new intelligent and complete solution to modern intrusion detection through the complete integration of each stage of the workflow into one optimized, unified deep learning framework. The final data output from this system goes through a rigorous process of preprocessing before becoming a valid representation of the data in the form of categories regarding information within the dataset, along with EDA statements are accompanied by EDA evidence. In addition, Shannon Entropy provides a better understanding of feature structure for the relevant projects by allowing the model to capture where there is variation in network behaviours and PCA provides a way to reduce dimensionality to enable more efficiency and eliminate redundant features. The knowledge obtained from both of these processes are then combined with the use of the FEA-CasSTFormerNet architecture using FEA, it allow the use of recursive attention and contextual data in real time to capture historical context in addition to the most recent framework. The combination of these then allow the architecture to model Deep Spatio-Temporal relationships of network traffic. Finally, Archerfish Hunting Optimization is also incorporated into this workflow to enable Fine Tuning of hyperparameters so as to optimally balance exploration vs. exploitation during the hyperparameter tuning process. Combined, these components produce a highly accurate, low false positive, and generalizable IDS capable of detecting the variety and changing nature of cyber-attacks. The proposed system is implemented in python and achieves an accuracy of 0.9998, precision of 0.9988, Recall of 0.9960 and F1-Score of 0.9974.

## VI. AUTHOR'S CONTRIBUTION

**Conceptualization:** S. Tamil Selvan, Haritha K Sivaraman, T. Anupama Thankam, and Gavaskar Vincent.

**Methodology:** S. Tamil Selvan, and Haritha K Sivaraman.

**Investigation:** S. Tamil Selvan, and Haritha K Sivaraman.

**Discussion of results:** S. Tamil Selvan, Haritha K Sivaraman, T. Anupama Thankam, and Gavaskar Vincent.

**Writing – Original Draft:** S. Tamil Selvan.

**Writing – Review and Editing:** S. Tamil Selvan, and Haritha K Sivaraman.

**Resources:** Haritha K Sivaraman.

**Supervision:** Haritha K Sivaraman, and T. Anupama Thankam.

**Approval of the final text:** S. Tamil Selvan, Haritha K Sivaraman, T. Anupama Thankam, and Gavaskar Vincent.

## VII. REFERENCES

- [1] R. Bensaid, N. Labraoui, A. A. Ari, H. Saidi, J. H. Emati, and L. Maglaras, "SA-FLIDS: secure and authenticated federated learning-based intelligent network intrusion detection system for smart healthcare," *PeerJ Computer Science*, vol. 10, pp. e2414, Dec. 2024.
- [2] R. Bhandari, S. Singla, P. Sharma, and S. S. Kang, "AINIS: An Intelligent Network Intrusion System," *International Journal of Performability Engineering*, vol. 20, no. 1, Jan. 2024.
- [3] K. Karthika, and D. P. Rangasamy, "Authorization of Aadhar data using Diffie Helman key with enhanced security concerns," *Journal of Intelligent & Fuzzy Systems*, Apr. 2024.
- [4] S. Duary, P. Choudhury, S. Mishra, V. Sharma, D. D. Rao, and A. P. Aderemi, "Cybersecurity threats detection in intelligent networks using predictive analytics approaches," In 2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM), pp. 1-5. IEEE, Feb. 2024.
- [5] O. Olabanjo, A. Wusu, E. Aigbokhan, O. Olabanjo, O. Afisi, and B. Akinuwesi, "A novel graph convolutional networks model for an intelligent network traffic analysis and classification," *International Journal of Information Technology*, pp. 1-13, Aug. 2024.
- [6] J. B. Pankiraj, V. Govindaraj, Y. Zhang, P. R. Murugan, and A. Milton, "Development of a scalable coding of encrypted images using Enhanced Block Truncation Code," *Webology*, vol. 19, no.1, pp. 1620-1639, Jan. 2022.
- [7] O. Arreche, T. R. Guntur, J. W. Roberts, and M. Abdallah, "E-xai: Evaluating black-box explainable ai frameworks for network intrusion detection," *IEEE Access*, vol. 12, pp. 23954-23988, Feb. 2024.
- [8] M. Hasanah, R.A. Putri, M. Aidie, R. Putra, and T. Ahmad, "Analysis of weight-based voting classifier for intrusion detection system," *International Journal of Intelligent Engineering and Systems*, vol.17, no. 2, pp. 190-200, Mar. 2024.
- [9] S. Al Rumhi, and J. Sivakumar, "Sales analysis-review and recommendations on business intelligence," In 2023 24th International Arab Conference on Information Technology (ACIT), pp. 1-10. Dec. 2023.
- [10] J. Yan, H. Xu, Z. Liu, Q. Li, K. Xu, M. Xu, and J. Wu, "{Brain-on-Switch}: Towards advanced intelligent network data plane via {NN-Driven} traffic analysis at {Line-Speed}," In 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), pp. 419-440. 2024.
- [11] A. A. Laghari, A. A. Khan, A. Ksibi, F. Hajje, N. Kryvinska, A. Almadhor, M. A. Mohamed, and S. Alsubai, "A novel and secure artificial intelligence enabled zero trust intrusion detection in industrial internet of things architecture," *Scientific Reports*, vol. 15, no. 1, pp. 26843, Jul. 2025.
- [12] M. Moukhafi, M. Tantaoui, I. Chana, and A. Bouazi, "Intelligent intrusion detection through deep autoencoder and stacked long short-term memory," *Int. J. Electr. Comput. Eng*, vol. 14, pp. 2908-2917, Jun. 2024.
- [13] D. Javeed, M. S. Saeed, M. Adil, P. Kumar, and A. Jolfaei, "A federated learning-based zero trust intrusion detection system for Internet of Things," *Ad Hoc Networks*, vol. 162, pp. 103540, Sep. 2024.
- [14] N. Girubagari, and T. N. Ravi, "Parallel ABILSTM and CBIGRU Ensemble Network Intrusion Detection System," *International Journal of Intelligent Engineering & Systems*, vol. 17, no. 1, Jan. 2024.
- [15] N. O. Aljehane, H. A. Mengash, S. B. Hassine, F. A. Alotaibi, A. S. Salama, and S. Abdelbagi, "Optimizing intrusion detection using intelligent feature selection with machine learning model," *Alexandria Engineering Journal*, vol. 91, pp. 39-49, Mar. 2024.
- [16] N. O. Aljehane, H. A. Mengash, S. B. Hassine, F. A. Alotaibi, A. S. Salama, and S. Abdelbagi, "Optimizing intrusion detection using intelligent feature selection with machine learning model," *Alexandria Engineering Journal*, vol. 91, pp. 39-49, Mar. 2024.
- [17] E. El-Shafeiy, W. M. Elsayed, H. Elwahsh, M. Alsabaan, M. I. Ibrahim, and G. F. Elhady, "Deep complex gated recurrent networks-based IoT network intrusion detection systems," *Sensors*, vol. 24, no. 18, pp. 5933, Sep. 2024.
- [18] H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci, "Large language models in wireless application design: In-context learning-enhanced automatic network intrusion detection," In GLOBECOM 2024-2024 IEEE Global Communications Conference, pp. 2479-2484, Dec. 2024.
- [19] Y. Li, J. Zhang, Y. Yan, Y. Lei, and C. Yin, "Enhancing network intrusion detection through the application of the dung beetle optimized fusion model," *IEEE Access*, vol. 12, pp. 9483-9496, Jan. 2024.
- [20] N. D. S. Nidamanuri, "AI techniques applied to network intrusion detection system by using genetic approaches," 2024.
- [21] H. Sharma, P. Kumar, and K. Sharma, "Recurrent neural network based incremental model for intrusion detection system in IoT," *Scalable Computing: Practice and Experience*, vol. 25, no. 5, pp. 3778-3795, Aug. 2024.
- [22] X. Ye, H. Cui, F. Luo, J. Wang, X. Xiong, W. Zhang, J. Yu, and W. Zhao, "Daily insider threat detection with hybrid TCN transformer architecture," *Scientific Reports*, vol. 15, no. 1, pp. 28590, Aug. 2025.
- [23] F. A. Vadhi, M. L. Salihi, and M. F. Nanne, "Machine learning-based intrusion detection system for detecting web attacks," *IAES International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 711-721, Mar. 2024.
- [24] N. Dash, S. Chakravarty, A. K. Rath, N. C. Giri, K. M. AboRas, and N. Gowtham, "An optimized LSTM-based deep learning model for anomaly network intrusion detection," *Scientific Reports*, vol. 15, no. 1, pp. 1554, Jan. 2025.
- [25] M. Shoab, and L. Alsbatin, "GRU Enabled Intrusion Detection System for IoT Environment with Swarm Optimization and Gaussian Random Forest Classification," *Computers, Materials & Continua*, vol. 81, no. 1, Oct. 2024.