Journal of Computer Science

Research Article

An Explainable AI (XAI) Framework for Intrusion Detection and Identification to Secure IoT Networks

Vibhor Harit[1], Rajeev Dahiya[1], Umang Garg[2] and Anil Kumar[3]

[1]*Department of Computer Science and Engineering, Desh Bhagat University Mandi Gobindgarh, Punjab, India*
[2]*School of Computer Science and Engineering, IILM University, Gurugram, Haryana, India*
[3]*Department of Computer Science and Engineering, Roorkee Institute of Technology, Roorkee, Uttarakhand, India*

Article history
Received: 26-02-2026
Revised: 11-06-2026
Accepted: 28-06-2026

Corresponding Author:
Vibhor Harit
Department of Computer Science and Engineering, Desh Bhagat University Mandi Gobindgarh, Punjab, India
Email: vibhordev@gmail.com

Abstract: The Internet of Things (IoT) encompasses a wide range of applications from wearable devices to smart homes, as well as industrial automation and smart cities. As IoT networks and devices continue to grow rapidly, security has become crucial. There are many techniques utilized to provide security aspects in IoT networks. These security aspects focus on mitigating risks and fixing vulnerabilities related to connected devices. Due to the sheer quantity, complexity, and amount of data, the increasing number of IoT devices does in fact increase the risk of new attacks and vulnerabilities. An intrusion detection system is supposed to be powerful for the detection and identification of attacks. Some intelligent models are trained with a set of features and evaluated by applying a filter-based approach for selecting significant features. Deep learning models have intricate architectures with many layers and neurons; they are sometimes seen as opaque and challenging to understand. Explainable Artificial Intelligence (XAI) methods have been devised to provide information about models. The utilization of XAI techniques, including Local Interpretable Model-agnostic Explanations (LIME) and Shapley Additive Explanations (SHAP), helps better understand the rationale behind model predictions and encourages the user to understand the model behavior better.

Keywords: IoT Security, Deep Learning, Intrusion Detection System (IDS), XAI, SHAP, LIME, Feature Selection

Introduction

IoT environments are heterogeneous in nature and comprise various devices interacting in a continuous manner and producing massive amounts of data. This heterogeneity and magnitude bring significant issues of data storage, big data processing, and security. With the recent significant growth of IoT networks, big data, blockchain, the Internet, and cloud computing, the security issue has become one of the most urgent needs. Therefore, IoT systems are quite susceptible to numerous types of attacks, and so the IDS should be effective to guarantee that the network remains reliable and safe (Zhukabayeva et al., 2025). IDS can monitor network activity, identify suspicious behavior, and send alerts where anomalies are detected. Some IDS methods are more focused on anomaly-based detection methods, in which models are trained to understand the normal system behavior and indicate abnormal practices as possible intrusions. AI and ML have been commonly employed to do this because they can learn intricate patterns using a vast amount of data. Nevertheless, it is essential to pay attention to the features that are selected prior to the use of ML or DL models (Rodríguez et al., 2025).

Feature reduction not only enhances model accuracy but also reduces computational complexity, storage, and training time, which is important in resource-constrained IoT settings. These models can be trained to gain high-level and hierarchical representations of network traffic data by using several steps of deep learning. These representations assist in capturing complex and subtle attack patterns that are not easy to identify using conventional machine learning methods, thus enhancing the detection accuracy and strength in the IoT. Most machine learning and deep learning models are mainly

limited by the fact that they are black boxes, although they perform very well. These models can be very hard to read internally, resulting in a lack of user confidence, especially on sensitive security applications that require one to know why a decision was reached.

To deal with this difficulty, XAI methods are presented. XAI is intended to enhance the level of transparency by showing the contributions that individual features make to the prediction by a model. XAI improves the level of trust, accountability, and usability of deep learning-based intrusion detection systems without negatively affecting their predictive capabilities. Models that can be used to enhance the interpretability of models include LIME and SHAP. These techniques assist researchers and practitioners in comprehending the reason behind a model declaring a network instance to be normal or malicious, hence boosting trust in the judgment of the system.

The presented work uses deep learning-based anomaly detection models, specifically a DNN, to categorize network traffic as normal and attack under a multiclass setup. A filter-based correlation feature selection method is used, wherein weakly correlated features are filtered out, and only those features that are strongly relevant are maintained (Naveeda and Fathima, 2025). This dimensionality reduction increases efficiency but does not provide accurate identification of attacks. Hyperparameter tuning and optimization are also performed for the optimization of the model. The proposed solution is an effective option to maximize the accuracy of intrusion detection and maintain interpretability with XAI, which is suitable to protect contemporary IoT networks. The contributions of the article under consideration are:

- To detect the anomaly in the IoT network by utilizing the DNN, 1D-CNN, and 2D-CNN models
- To categorize the features based on the embedded approach that integrates a correlation-based filter and a wrapper method
- To compare and evaluate the existing system with the proposed method

Literature Review

Recognized literature reflects the increasing usefulness of DL methods in IDS on IoT networks. These articles investigate numerous DL models, simulations, and benchmarking data, which prove the adaptability of DL models to deal with complex and dynamic IoT security threats. The IDS based on DNN proposed by G. Thamilarasu et al. was developed with the assistance of the Keras library (Thamilarasu and Chawla, 2019). The model was composed of three hidden layers with ReLU activation functions, and it was trained for 150 epochs. The system was tested through the Cooja network simulator to make sure it is realistic and appropriate to the environment of IoT. The Cooja network simulator is specifically designed to simulate IoT network environments using the Contiki operating system. The results of the experiment revealed that there were good detection rates of various attack types, with a precision level of 95.7% on Opportunistic attacks, 96% on DDoS attacks, 99.5% on Sinkhole attacks, and 96% on Wormhole attacks. These findings clearly showed that the proposed DL-based IDS was much more effective than the conventional intrusion detection methods that used inverse weight clustering, which confirmed the usefulness of deep learning in the application of IoT security solutions.

A. Nagisetty et al. observed the effectiveness of four types of deep learning models, namely CNN, DNN, Multi-Layer Perceptrons (MLP), and autoencoders, in terms of intrusion detection in IoT networks (Nagisetty and Gupta, 2019). The models were coded in Python on Keras and tested on two popular benchmark datasets, UNSW-NB15 and NSL-KDD99. The experimental results indicated that the MLP model had the best F1-scores, with a score of 95.76% on the UNSW-NB15 dataset and 99.28% on the NSL-KDD99 dataset. Such results also support the possibility of DL-based methods to enhance the accuracy and reliability of intrusion detection in IoT settings. DNN architecture has the best overall accuracy value of all the tested models, ranging to 99.24%, which is an excellent sign of its strength and usefulness as a tool in intrusion detection.

Qiu et al. examined adversarial attacks on IoT settings of NIDS that are based on DL. Their results indicated that minimal changes in the characteristics of packets would greatly manipulate model estimates, which could support denial-of-service (DoS) attacks (Qiu et al., 2020). The Kitsune NIDS was trained on PyTorch, and experiments were run on a Feature Mapping (FM) model that was trained on 10 percent of the data and an anomaly detection model that was sampled in the Mirai botnet. The initial learning rate was 0.1 with the Stochastic Gradient Descent (SGD) optimizer, and the adversarial experiments achieved an attack success rate of more than 95, indicating the susceptibility of the DL-based IDS to an adversarial input.

The framework presented by Meidan et al. (2018) is called N-BaIoT, and it is an anomaly detection framework developed with an explicit focus towards the usage of deep autoencoders based on IoT networks. The network traffic of two major IoT botnets, BASHLITE and Mirai, was used in training the model. The autoencoder model was based on four encoding hidden layers that had gradually smaller dimensions (75, 50,33, and 25, the size of the input). The Keras framework was used to carry out hyperparameter tuning. The findings indicated that the attacks were identified in 174 milliseconds with a True Positive Rate (TPR) of 100 percent and a very low

average false positive rate (FPR) of 0.007, which indicated the effectiveness and precision of the method.

Vinayakumar et al. (2020) suggested a DL-enhanced botnet detection model that targets the DNS traffic examination in IoT networks. The visualization techniques used in the study include advanced visualization of feature embeddings and the characteristics of the dataset. The domain names obtained in DNS logs were encoded at the character level with Keras, and the results were converted to vectors. Then, these vectors were input into CNN and RNN models, which were optimized with the Adam optimizer on the DMD-2018 dataset. The given deep learning models scored an F1-score of 0.916 and an overall accuracy of 99 percent, which can be regarded as the clear manifestation of the deep learning methods' applicability in the context of botnet activity detection in an IoT network. These findings support the fact that deep learning models can be trained to learn intricate traffic models and differentiate between abnormal and normal network behavior with a high degree of accuracy. The presence of high accuracy and a high F1 score demonstrates the overall balanced performance, i.e., the presence of low false positives and a low false negative, which is essential in real-life intrusion detection systems.

Moreover, Sun et al. (2020) presented a hybrid deep learning-based intrusion detection system comprising the CNN and LSTM framework. In such a design, CNN layers oversee deriving spatial and local data of network traffic data, and LSTM layers are tasked with identifying temporal dependencies and sequential patterns of traffic flows. To solve the problem of class imbalance, a weight optimization strategy was implemented, where the weights of the classes were calculated according to the distribution of samples. This hybrid architecture is much more accurate and robust in terms of detection and is highly suitable in the IoT intrusion detection setting. The dataset to test the method was CICIDS2017, which got an accuracy of 98.67 and 99.5% when applied to each of the types of attack (Potharaju et al., 2025). The explainable AI approaches offer local and global explanations. Global explanation looks at the model predictions of all the variables, and local explanation focuses on explaining certain predictions. Whereas a local explanation accounts for the individual prediction, a global one accounts for the contribution of all features to the decision and model prediction. Local explanation contributes to the believability of the model by identifying a certain case and attributing (Mohammed et al., 2025). To build an effective model and perform performance evaluation by decision-making and optimizing parameters. There are two types of XAI techniques (Benmalek and Seddiki, 2025) that describe the trained model's behavior. Instead of changing the model, the explanation will look at its predictions and boost confidence in it. XAI techniques provide an explanation of the model, which is then modified without compromising its interpretability and explainability (Yazdinejad et al., 2024). Many sectors employ explainable AI to provide an explanation for model predictions. For instance, researchers suggested using stable LIME to explain models in the healthcare system. The model is built using random forest classification for the identification of breast cancer with higher accuracy (Kasongo and Sun, 2020).

Materials and Methods

The proposed methodology is segregated into distinct modules that are shown in Figure 1. The modules include several steps, such as data preprocessing for normalizing & encoding the data, feature selection for identifying the significant features, selection of important features with inputs for transforming the data, and model training for validating the results with the test data. The standard datasets are used to run the model on Google Colab. There are two datasets used in the current research: NSL-KDDnew and UNSW-NB15new. The NSL-KDDnew dataset is an improved version of the KDD Cup 1999 dataset designed for intrusion detection research. It removes redundant records and provides balanced training and testing sets. The dataset contains normal and attack traffic categorized into DoS, Probe, R2L, and U2R classes, making it widely used for benchmarking IDS models. The UNSW-NB15new dataset was developed to represent modern network traffic and attack scenarios. It includes normal traffic and nine attack categories such as Fuzzers, Exploits, and Worms. It is generated by using realistic network simulations with 49 features.

Data Preprocessing

It eliminates superfluous data from the datasets after uploading them to Google Colab and then encodes the features to convert nominal range to integer range values. The normalization technique is followed by the encoding technique for finding the normalized value range. The characteristics are converted into integer values using a variety of encoding methods, including label and one-hot encoding. To standardize the datasets and enhance the performance and other parameters of the model, the data must be normalized once it has been encoded, such that the data is between 0 and 1. The normalization utilized was Min-Max, which normalizes the value between 0 and 1 (Bae et al., 2019). Once the dataset was extracted, it transferred the CSV file format in the local system to the Collaboratory system at Google and read the dataset into a Pandas DataFrame. It converted the dataset's category features to numerical values after removing unnecessary columns. Two encoding techniques are commonly used: label encoding and one-hot encoding. In this work, label encoding was applied to transform categorical feature values into integer representations.

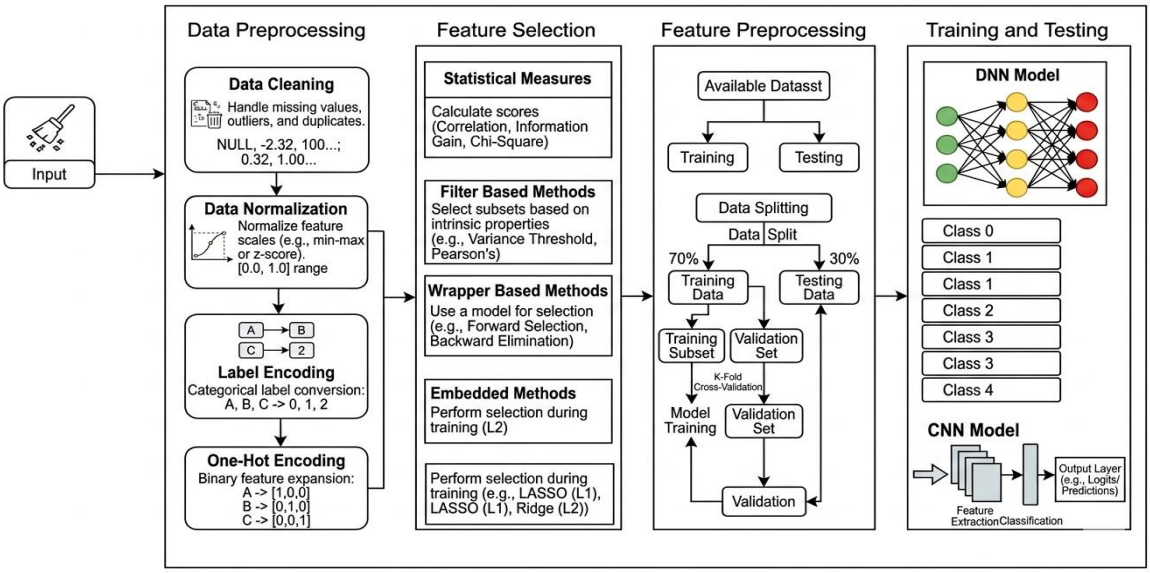


Fig. 1: Proposed Methodology of the system

Since the dataset contains features with different value ranges, normalization was performed to scale all values between 0 and 1. Protocol type, flag, services, and class are the four characteristics in the NSL-KDD new dataset that include categorical values (3, 70, 11, and 5 categories, respectively). The four categorical value features of UNSW-NB15 are proto, service, status, and attack cat, with corresponding categories of 133, 13, 11, and 10 (Ferrag et al., 2020). It employed the label encoding approach to encode the category data using integer values. The dataset was normalized to a common scale using the Min-max normalization (Eq. 1):

$$F_{new} = \left(F - F_{min}\right) / \left(F_{max} - F_{min}\right) \quad (1)$$

Where F_{min} and F_{max} are the feature's smallest and maximum values, respectively, and F is the feature's actual value in the column. Additionally, F_{new}, the new value, falls between 0 and 1.

Feature Selection

Feature selection strategies limit the number of inputs to the model, which in turn lowers the computational and storage costs. Features are chosen prior to model training because the filter-based approach requires less time and processing. The strongly correlated characteristics with threshold values greater than 0.95 are chosen using the PCC correlation approach (Alavizadeh et al., 2022). The most important feature is selected by using recursive feature elimination. The most correlated features can be identified by using the prominent features as input for model training. The correlation-based filter technique is utilized for the enhancement of model performance and reduces computational time. The selection of features using the correlation-based filter method depends on the correlation score to select features. It uses the Pearson correlation method to determine how the features are related to each other. The similarity index can be identified by identifying the correlation between two variables, such as V1 and V2, which can be written as (Eq. 2):

$$PCC = \text{cov}\left(V_1, V_2\right) / \sigma\left(V_1\right) \sigma\left(V_2\right), \quad (2)$$

The equation is utilized to evaluate the Pearson Correlation Coefficient (PCC) in terms of covariance (cov) that lies between -1 and 1 and the standard deviation of two variables (V1 and V2). The PCC value of highly correlated features is close to −1 and 1, whereas that of uncorrelated features is close to 0 (Olanrewaju-George and Pranggono, 2025). The Pearson correlation is used to choose features during the model-building process. In the new dataset, 36 features have been removed. Some features are removed from the dataset to establish correlation among the features. As a result, 38 features were kept in the new dataset. There are several features removed from the NSL-KDD Cup dataset, such as num root, srv serror rate, dst host sserver rate, dst host sserver serror rate, and srv rerror rate.

The heat map visualization of the correlation of the new NSL-KDD and UNSW-NB15 datasets highlights the most highly associated features, which are shown in Figures 2 and 3. Features that are considered redundant and have a value higher than the 0.95 criterion are chosen, and the redundant feature is removed by dropping one of the features.

DOI: 10.3844/jcssp.2026.2274.2290

Fig. 2: Heat map visualization of highly correlated features for NSL-KDD Dataset

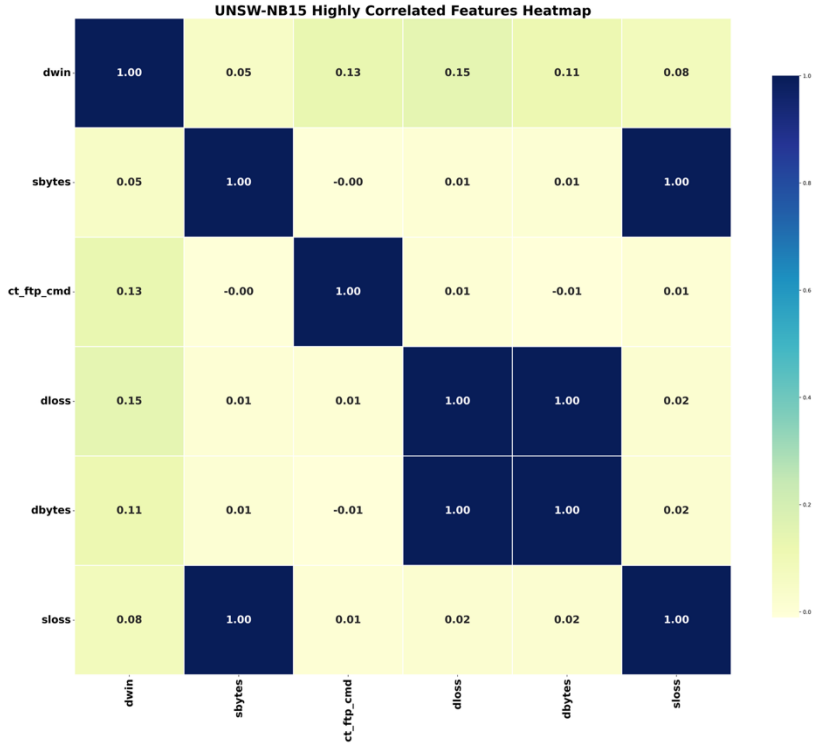


Fig. 3: Heat map visualization of highly correlated features for UNSW-NB 15 Dataset

To minimize redundancy, one of the highly redundant features is excluded from the set of features with high correlation values based on the threshold value. Features were considered redundant if their correlation values were higher than the cutoff. Next, one of the highly associated traits was removed. Then one of the most associated characteristics was eliminated. Features with correlation values greater than the criterion of 0.95 were selected and deemed redundant. Next, one feature was eliminated from the set of unnecessary features.

Feature Preprocessing

The dataset is then transformed into a format suitable with the model so that it can be trained. The data is prepared for use in the DNN model and is immediately trained in the model upon feature selection. The dataset is transformed into a suitable format prior to the CNN models being trained. The dataset in 1D vector form can be utilized with the 1D CNN model. In the 2D-CNN model, the transformation of the dataset is conducted with 2D vectors, and then nxn matrices are transformed based on the features.

The selected features are used as input to the models so that the processed data becomes compatible with the model, and it is trained after encoding and normalization. The DNN model uses datasets, and these datasets are compatible with it. The data is converted to a vector form so that it can be input into the CNN model. The data is fed into the 1D CNN model as an input, which has been turned into a 1D vector format. The 2D CNN model is trained with the number of features under its input into a 2D matrix and then processed through the model. This is converted into an n x m vector, and the dataset is padded with 0 before being converted to the closest perfect square if the features are not perfect squares. It first converts the 36 features in the NSL-KDD new dataset into a 6 x 6 matrix before supplying the inputs to the model. Then every 36-feature record is converted into a 6 x 6 matrix image. Similarly, the 38 features of the UNSW-NBnew dataset are transformed into a 777 pixel-padded 7×7 matrix.

Training and Testing

The training of the model is conducted with 75% of the dataset, while 25% of the dataset will be utilized for testing purposes. Out of the 75% training dataset, 15% of the dataset is reserved for validation purposes. After applying the training dataset to the deep learning training model, the model is validated using the validation dataset and categorized as either normal or attack class type (Al-Haija and Droos, 2024). Convolution and max-pooling layers are used to construct the CNN model, whereas dense hidden layers are used to construct the DNN model. The models are then trained and evaluated. To prevent overfitting, the model is tuned using a range of parameters and hyperparameters.

Model Building

This describes the model's forecast once it has been fully trained and tested. Using a testing dataset, the explainable AI notion is applied to model prediction. SHAP and LIME are two popular surrogate models that help comprehend the ideas. Model-specific and model-agnostic are the two main types of model explainability. The presented methodology involves XAI with deep learning models, which guarantee predictive accuracy and interpretability. The XAI methods can be generally divided into model-agnostic and model-specific methods. Unlike model-specific approaches that require a specific model, like a linear regression, a decision tree, or a neural network, model-agnostic methods can be applied to every machine learning model by looking at the inputs and outputs. The most common tools of XAI are LIME and SHAP. LIME offers some local explanations, trying to explain the behavior of the model near a given prediction, which helps to view the reason why a certain instance was deemed normal or an attack. SHAP, however, provides both local and global explanations and quantifies both the contribution of each feature to single predictions and the behavior of the model in general (Ahmad et al., 2025). Preprocessing of the data includes encoding, scaling, and transformation, and then it is input into the deep learning models. The processed data is further divided into training, validation, and testing subsets randomly. Training and validation are usually done on 75 percent of the data, whereas 25 percent of the data is set aside to test the model, as shown in Figure 4. Learning structures include DNN and CNN. The DNN employs the use of several dense hidden layers, which are made up of connected neurons to learn complicated patterns. Conversely, the CNN involves convolutional layers that retrieve feature maps of the input data and then pooling layers, i.e., max-pooling and average-pooling, to reduce the dimensions (Gulzar and Mustafa, 2025). These features obtained are then fed to fully connected layers to be classified.

In the training process, the model parameters are adjusted repeatedly to minimize the loss and enhance the performance. The trained models are tested on test data that is not seen, and it has normal and attack cases. The fact that the training and testing accuracy are similar is a sign of good generalization, whereas testing accuracy that is high and training accuracy that is low is an indication of overfitting. This scale assessment can be used to guarantee its sound performance in the real world and its interpretability using XAI methods.

Data Set Description

This phase evaluated the model by taking two publicly available databases, NSL-KDD Cup and UNSW-NB15.

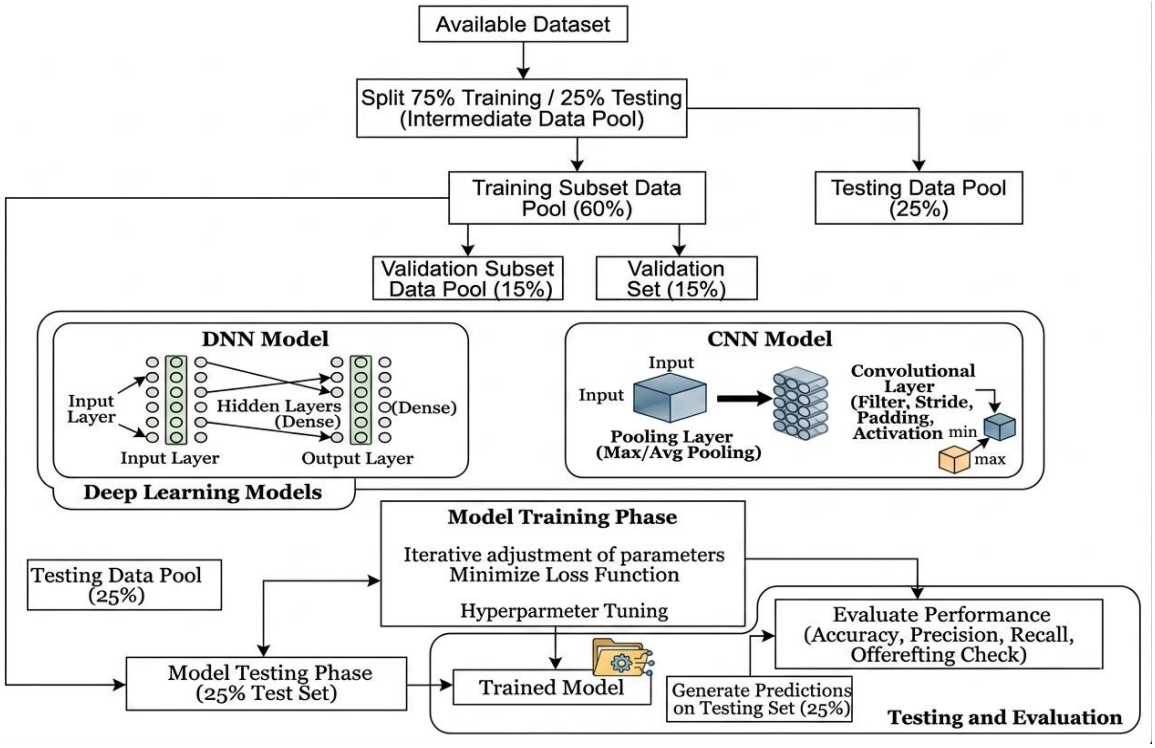


Fig. 4: Feature Selection Model

The major weakness of the KDDCup data set, obtained through the NIDS Evaluation Program of DARPA funded by MIT Lincoln Labs and used extensively in the NIDS area, is the occurrence of redundancy or duplication of information. This resulted in the creation of the NSL-KDD data set, where there is no repeated or duplicate data. A total of 42 features are represented (38 numerical, 3 nominal, and 1 label that represents the normal/attack category) (Borji, 2019). The dataset has 23 types of attacks, but it restricted the types of the assault classes by dividing them into four key classes of attack, that is, Probe, DoS, U2R, and R2L. In Table 1, the dataset has 125972 records, of which 67342 belong to the normal category, and the other classes of the attack are 11656, 995, 45927, and 52 in the Probe, R2L, DoS, and U2R categories, respectively. The statistics of the NSL-KDD Cup data can be seen in Table 2. The models are compared on the UNSW-NB15 dataset, which is a popular benchmark dataset designed by the Australian Centre of Cyber Security (ACCS) laboratory to carry out network intrusion detection studies. This data is intended to capture the current trend of network traffic and comprises both benign and malicious traffic, which would be suitable for the evaluation of the strength of a deep learning-based security model. There are a total of 44 features in the UNSW-NB15 dataset. Besides that, the dataset contains four nominal or categorical variables that are discrete values that describe characteristics like protocol type, service, or connection state (Rahman et al., 2025). The last feature is the label column that is used to declare whether a network instance is in normal operation or an attack.

Table 1: Total records with attack categories in NSL-KDD

Attack Class	Total Records
U2R	52
R2L	995
Probe	11,656
DoS	45,927
Normal	67,343

Table 2: Classification of Attacks

Attack Type	Attack Sub-Type	No. of Records
Attack	DoS	16,353
Attack	Exploits	44,525
Attack	Backdoor	2,329
Attack	Analysis	2,677
Attack	Fuzzers	24,246
Normal	Normal	93,000
Attack	Generic	58,871
Attack	Worms	174
Attack	Shellcode	1,511
Attack	Total Attacks	164,673
Attack	Reconnaissance	13,987
Overall	Total Records	257,673

The label feature is a multiclass feature with ten different classes, one of which is a normal one, and the rest are attack categories. The attack classes are generic attacks, reconnaissance, DoS, analysis, worms, shellcode, backdoors, and fuzzers. These varied types of attacks represent a broad scope of intrusion detection in the real world.

Through this dataset, the models are compared on binary (normal vs. attack) and multiclass classification tasks to enable a deeper assessment of their detection capability, generalization performance, and their application in real-world network security settings. It consists of 93000 records in the normal class out of the dataset's total of 164673 attack classifications (Hairani et al., 2024). There are 257673 records in the collection. Four attack types- Generic, Exploits, Denial-of-Service, and Fuzzers as well as five classes of common attacks were selected for experimental evaluation. The problem of class imbalance was addressed by undersampling the dataset. It selected 50K-sized samples at random for classes having more than 50K records, while it selected all records for classes with fewer than 50K records (Maniatopoulos and Mitianoudis, 2021). Then, it randomly chose 50K-sized samples from the generic and normal classification, which is greater than the sample size (Moustafa and Slay, 2015). Then it selected all records for the Fuzzers, DoS, and Exploits classes since their records were smaller than the sample size.

Results and Analysis

Once the dataset has been loaded into Google Colaboratory, it is preprocessed to standardize the dataset and encode it. The second step of the methodology is dedicated to the selection of the features to determine the most useful characteristics that render the dataset appropriate and effective to be used in training the model. The model complexity is reduced, and the learning efficiency and prediction accuracy are enhanced by eliminating irrelevant or redundant features. After the best feature combination has been established, the entire dataset is split into three subsets to facilitate both training and testing. It assigns 60% of the data to the training set, with the help of which the model parameters are learned. The remaining 15 percent of the data constitutes the validation set, and it is used to help tune the hyperparameters, track learning progress, and avoid overfitting in training. The other 25 percent of the data is then held as the testing set and is not utilized until the completion of the training to check how the model will perform on the unknown data.

The models are trained with the help of the TensorFlow deep learning platform. The DNN structure incorporates four dense hidden layers that help the model to adapt to intricate, nonlinear patterns in the chosen features and enhance the classification accuracy. The final dense layer includes five neurons, whereas the earlier three layers consist of 64 neurons and the ReLu activation function, depending on the classes. By comparing the likelihood of the actual and expected values, the loss, that is, the difference between the actual and predicted values, is computed. The model training procedure is aimed at loss minimization. To avoid overfitting the model, it utilizes diverse hyperparameters such as weight decay and dropout rate.

The design of the architecture and the hyperparameter optimization and performance analysis of the proposed deep learning models will be described as shown in Figure 5. Three convolutional layers comprising 64, 32, and 32 neurons are developed to form a 2D-CNN. All convolutional layers have a (3, 3) kernel and the ReLU nonlinear activation. A pooling layer of size (2, 2) is used to reduce spatial dimensionality and computational complexity. The last fully connected layer has the number of classes within the dataset. The number of classes in the dataset is five, and therefore the output layer is made of five neurons and uses the SoftMax activation function to produce the probability distributions of the classes. Besides the 2D-CNN, a 1D-CNN model is also carried out, in which the kernel size and pool size are three and two, respectively. Both CNN models are also optimized by changing some important hyperparameters, such as the kernel size and dropout rate, to enhance generalization and stability. To reduce overfitting, various sets of hyperparameters are tested with a dropout rate of 0.01, a weight decay of 0.0001, and a learning rate of 0.001. The models undergo 20 training iterations. Independent test sets are used to evaluate modeling performance. The models achieved 0.992, 0.993, and 0.994 accuracy for 1D-CNN, DNN, and 2D-CNN models. The models perform at accuracy values of 0.80 when applied to the UNSW-NB new dataset with DNN, 1D-CNN, and 2D-CNN. Lastly, the models are tested on the previously unknown data so that their performance on generalization in real-world situations can be measured after 20 training epochs.

Confusion Matrix

It makes it easier to visualize the output of the model by presenting the various results in a tabular form. In this section, the sensitivity of the proposed deep learning models will be assessed and compared by comparing the standard classification measures of two benchmark datasets, NSL-KDDnew and UNSW-NBnew. The first step involves constructing confusion matrices of both datasets to visualize the classification performance of DNN, 1D-CNN, and 2D-CNN models. The elements on the diagonal of the confusion matrix are TP, which are the samples that are classified correctly in each class; the off-diagonal elements correspond to misclassifications. Figures 6 and 7 represent these confusion matrices of the NSL-KDDnew and UNSW-NBnew testing datasets, respectively, which can be used to familiarize oneself with the behavior of the classes in prediction.

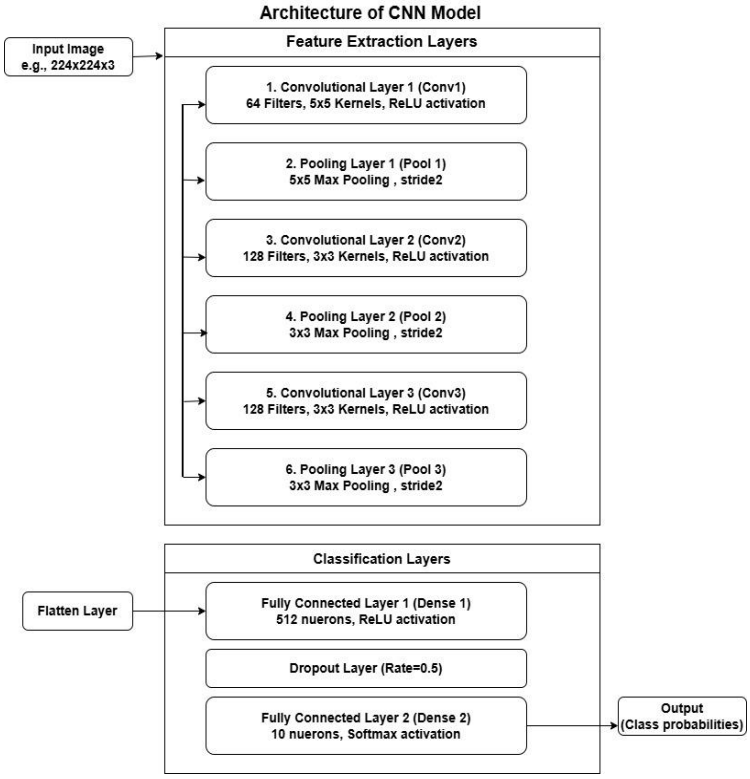


Fig. 5: CNN Architecture

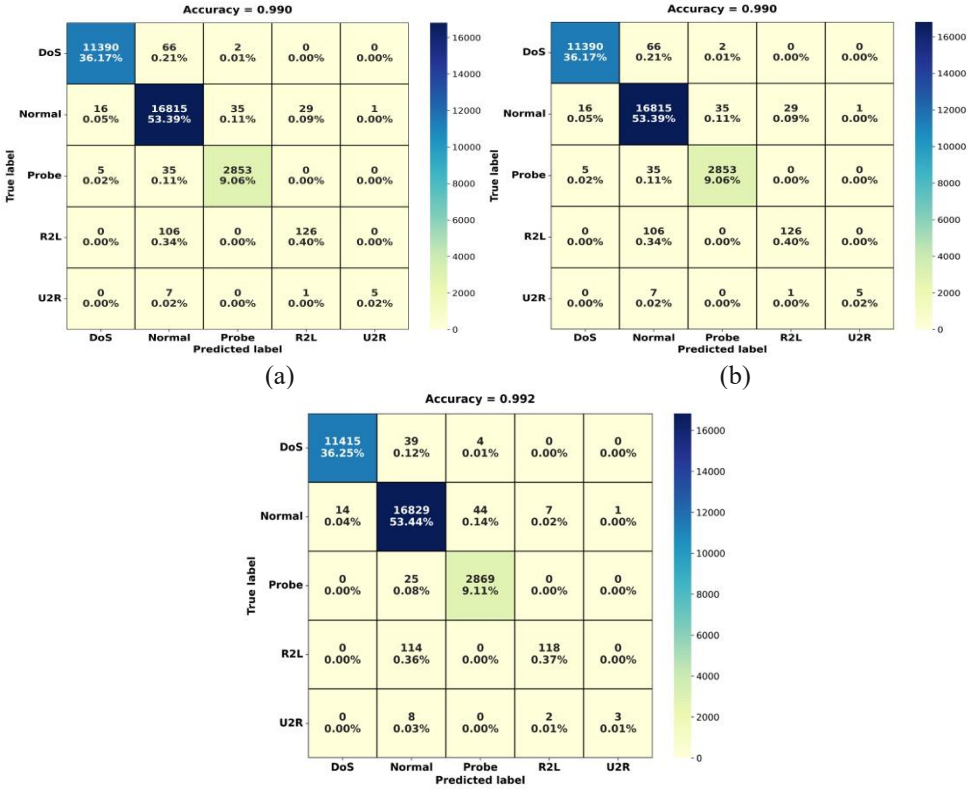


Fig. 6: Confusion matrix for NSL-KDDnew Dataset with (a) 1D-CNN (b) DNN (c) 2D-CNN Models

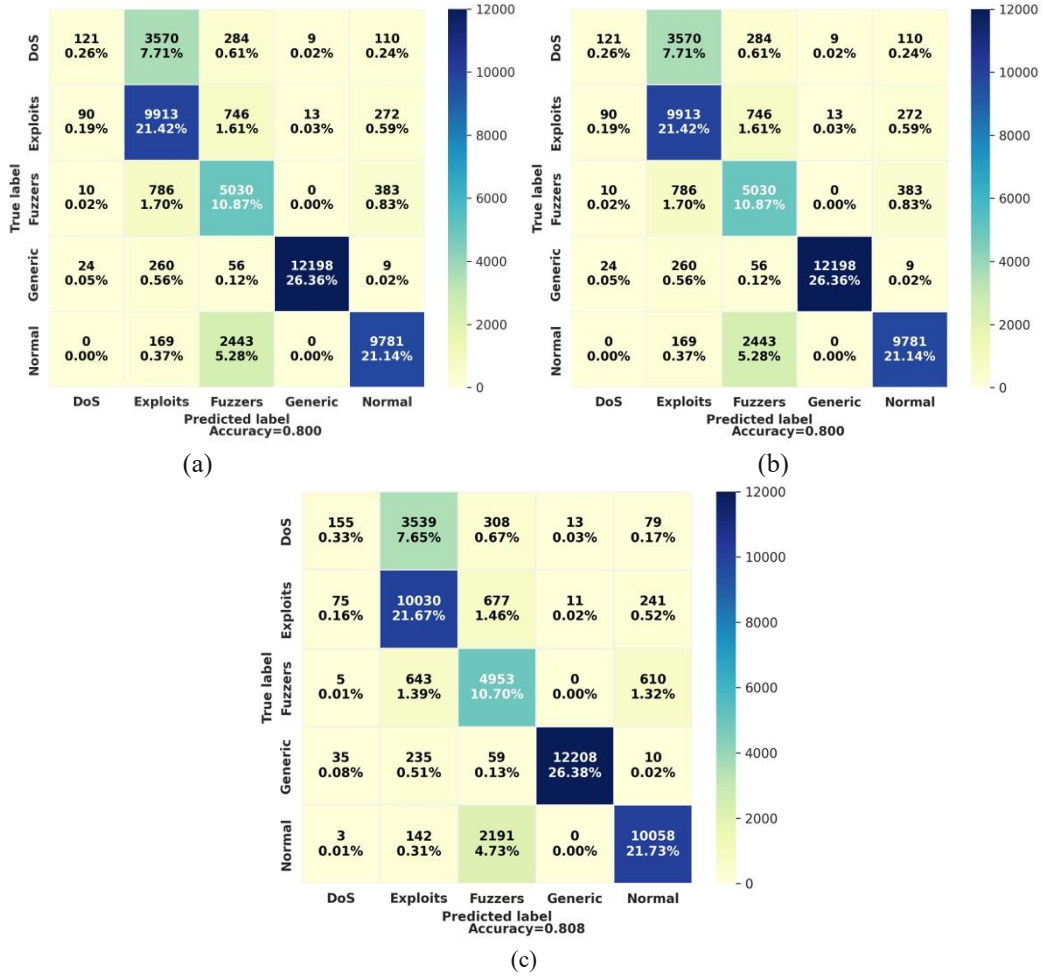


Fig. 7: Confusion matrix for UNSW-NBnew Dataset with (a) 1D-CNN (b) DNN (c) 2D CNN Models

To assess performance, P, R, and F1-Score are computed. Precision quantifies the number of predicted attacks that are attacks, and as such, it shows the decrease in FP. Recall shows the percentage of actual attack examples that are correctly identified, which reduces FN. The F1-Score is a combination of precision and recall, and it is a strong measure of unbalanced datasets. These measurements are calculated in NSL-KDDnew (Table 3) and UNSW-NBnew (Table 4), and their comparative findings are represented as Figures 8 and 9.

Accuracy and Loss

These findings indicate good performance on the NSL-KDDnew dataset. The history of the proposed DNN model gives an accuracy of 0.99 with a small value of loss at 0.04 against a learning rate of 0.001, weight decay of 0.0001, dropout of 0.01, and 20 training epochs. The 1D-CNN model has the lowest loss of 0.02 with an accuracy of 0.989, and the 2D-CNN model gives the best result with a loss of 0.02 as well as having an accuracy of 0.994 under the same training conditions. Figure 10 shows the accuracy Vs epochs for NSL-KDD and UNSEnew datasets. This denotes the better performance of the 2D-CNN model on this data. In the more complicated UNSW-NBnew offence, there is less performance because of increased variability of data. The accuracy of the DNN model is 0.80 with a loss of 0.47; the 1D-CNN is 0.80 with a loss of 0.41; and the 2D-CNN is slightly higher with 0.81 and 0.40, respectively. In general, the findings indicate that, even though all the models are generalized well, the 2D-CNN is always able to offer superior accuracy-loss trade-offs across datasets.

Training Time of Model

This part compares the performance and computational efficiency of the proposed deep learning models. Training time is calculated by subtracting the beginning and ending times of the model in the process of learning. On the NSL-KDDnew dataset, training on the DNN took 142 ms, 1D-CNN took 325 ms, and 2D-CNN took 340 ms.

Table 3: Performance Parameters with DNN, 1D-CNN and 2D-CNN model for NSL- KDDnew Dataset

Model / Metric	DoS (P)	DoS (R)	DoS (F1)	Normal (P)	Normal (R)	Normal (F1)	Probe (P)	Probe (R)	Probe (F1)	R2L (P)	R2L (R)	R2L (F1)	U2R (P)	U2R (R)	U2R (F1)	Accuracy
DNN	1.00	0.99	1.00	0.99	1.00	0.99	0.99	0.99	0.99	0.80	0.54	0.60	0.83	0.38	0.53	0.99
1D CNN	1.00	0.99	0.99	0.98	0.99	0.99	0.98	0.99	0.98	0.85	0.70	0.77	0.58	0.54	0.56	0.99
2D CNN	1.00	1.00	1.00	0.99	1.00	0.99	0.98	0.99	0.99	0.93	0.51	0.66	0.60	0.23	0.33	0.99

Table 4: Performance Parameters with DNN, 1D-CNN and 2D-CNN model for UNSW- NBnew Dataset

Model	DoS (P)	DoS (R)	DoS (F1)	Exploits (P)	Exploits (F1)	Exploits (R)	Fuzzers (P)	Fuzzers (F1)	Fuzzers (R)	Generic (P)	Generic (R)	Generic (F1)	Normal (P)	Normal (R)	Normal (F1)	Accuracy
DNN	0.49	0.03	0.06	0.67	0.77	0.90	0.59	0.68	0.81	1.00	0.97	0.98	0.93	0.79	0.85	0.80
1D CNN	0.48	0.06	0.10	0.64	0.76	0.94	0.65	0.66	0.67	0.99	0.97	0.98	0.91	0.81	0.85	0.80
2D CNN	0.57	0.04	0.07	0.69	0.78	0.91	0.60	0.69	0.80	1.00	0.97	0.99	0.91	0.81	0.86	0.81

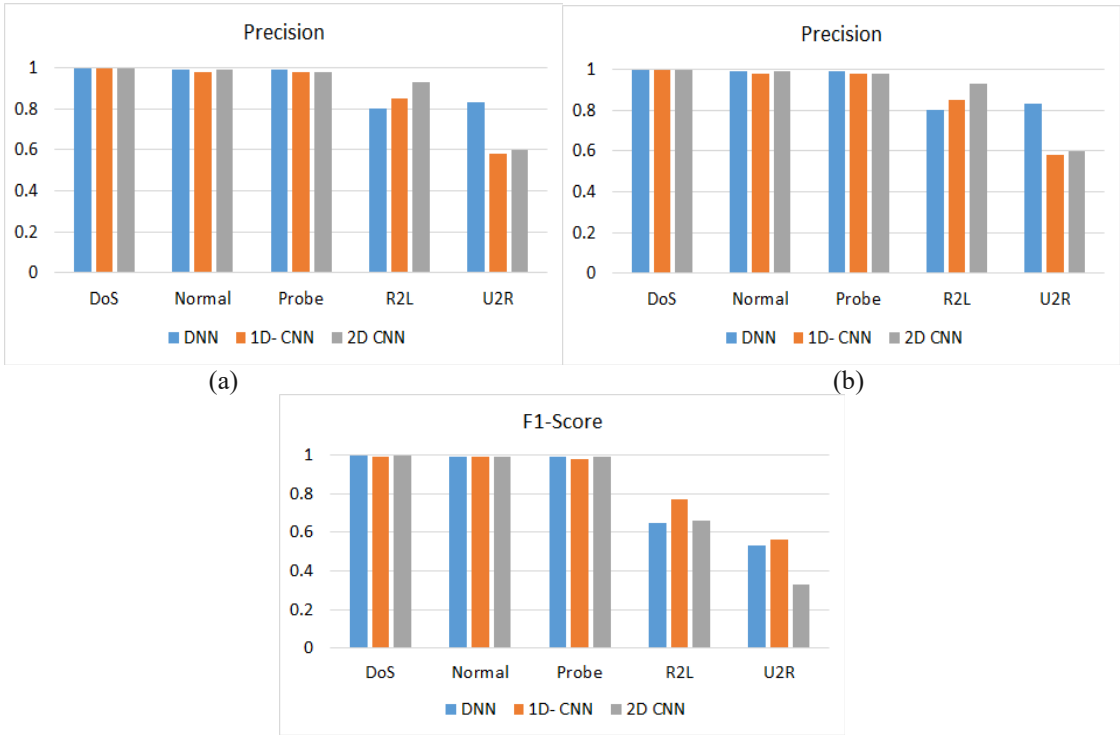
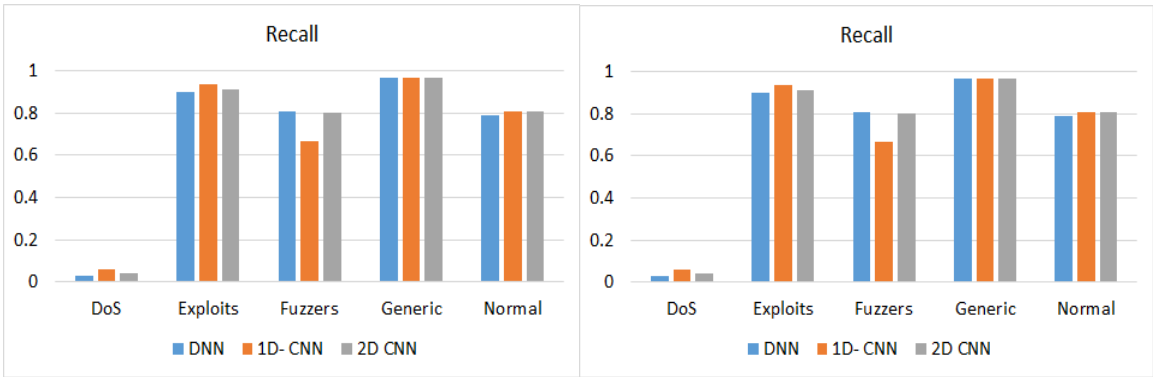


Fig. 8: DNN, 1D-CNN and 2D- CNN model parameters (a) Precision, (b) Recall and (c) F1-Score for NSL-KDDnew dataset



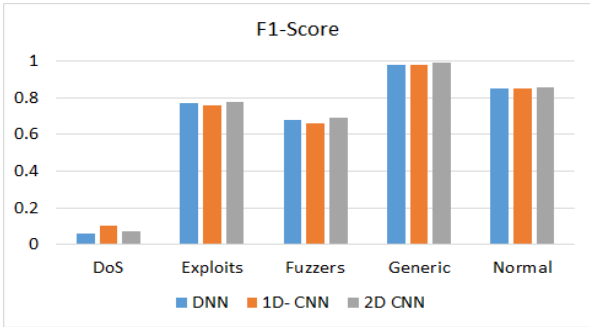


Fig. 9: Performance parameters for DNN, 1D-CNN, 2D-CNN models (a) Precision (b) Recall (c) F1-Score

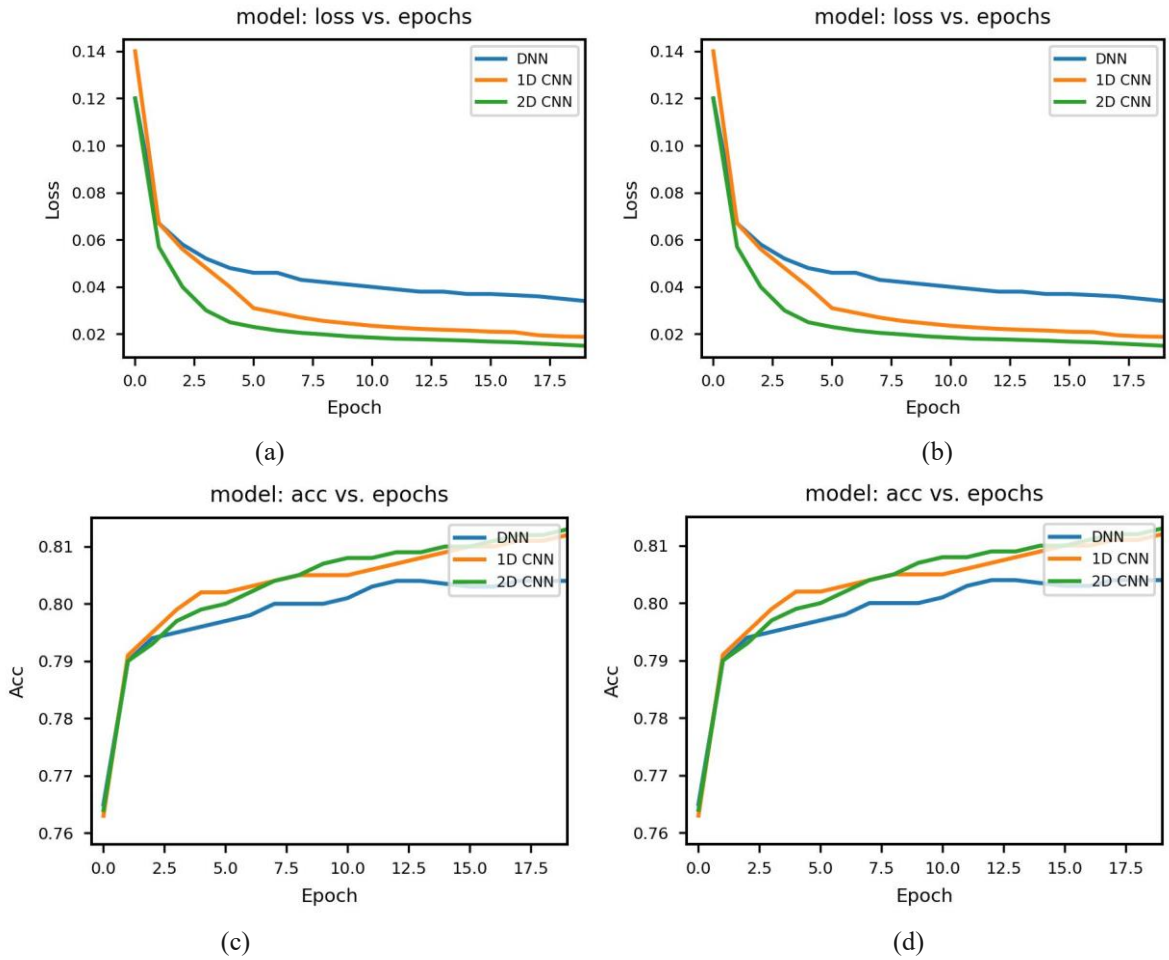


Fig. 10: Accuracy vs. Epochs for (a) NSL-KDDnew (b) UNSW-NBnew dataset and Loss vs Epochs for (c) NSL-KDDnew dataset (d) UNSW-NBnew dataset

On the same note, in the case of the UNSW-NBnew data, the DNN was trained in 323 ms, and 1D-CNN as well as 2D-CNN took 442 ms and 455 ms, respectively. Such findings suggest that the CNN-based models (especially the 2D-CNN) are less expensive in terms of computation and training. Conversely, the DNN model is always found to be faster in training both data sets. The 2D-CNN model is also more efficient in terms of accuracy and robustness than the DNN despite the increased training time. Here, the conflict between computational efficiency and predictive performance is indicated. The findings of the two datasets validate a point of view that

even though CNN architectures require more resources, they are in a better position to capture spatial and hierarchical feature representations, which translate into increased accuracy in the classification.

Discussion

To compare the current research with other models such as Decision Trees (DT), Support Vector Machines (SVM), and k-Nearest Neighbors (KNN), each model is trained and evaluated on the same preprocessing pipeline consisting of label encoding, normalization, and feature selection to have a fair comparison. As shown in Figure 11, the results indicate that deep learning models, specifically the 2D-CNN, perform better than classical machine learning models in terms of accuracy at multi-class classification on both datasets. Moreover, the research indicates that machine learning and deep learning models can be used in hybrid versions that can guarantee greater accuracy with less complex training. Table 5 represents the proposed models with the existing literature, such that the suggested 2D-CNN model can become competitive or even better, but with more training time needs. XAI models offer explanations both at a local and global scale. Global explanation examines the model predictions for every variable, whereas local explanation concentrates on interpreting specific predictions. Whereas a local explanation can explain a single prediction, a global explanation will use all features to explain the decision and the prediction made by the model.

It shows the performance of the model with decision-taking abilities & explanation. These techniques help to both improve the model's performance and comprehend its behavior. Figure 12 shows the interpretability results of the proposed model by using SHAP and LIME XAI techniques. SHAP can be used to explain both globally, by identifying the most influential features controlling model predictions, and locally, by modeling individual predictions. Based on the feature importance graph, the most important features for the classification of attacks are src_bytes, dst_bytes, count, and duration. SHAP force graph shows the effect of each feature value on one prediction. LIME provides local interpretation by showing the contribution of all the features to the chosen instance, both positive and negative. The feature importance and probability plots reveal that most of the attributes used to predict an attack are related to network traffic. Results illustrate that the framework is robust in terms of high detection accuracy and meaningful explanations for model decisions.

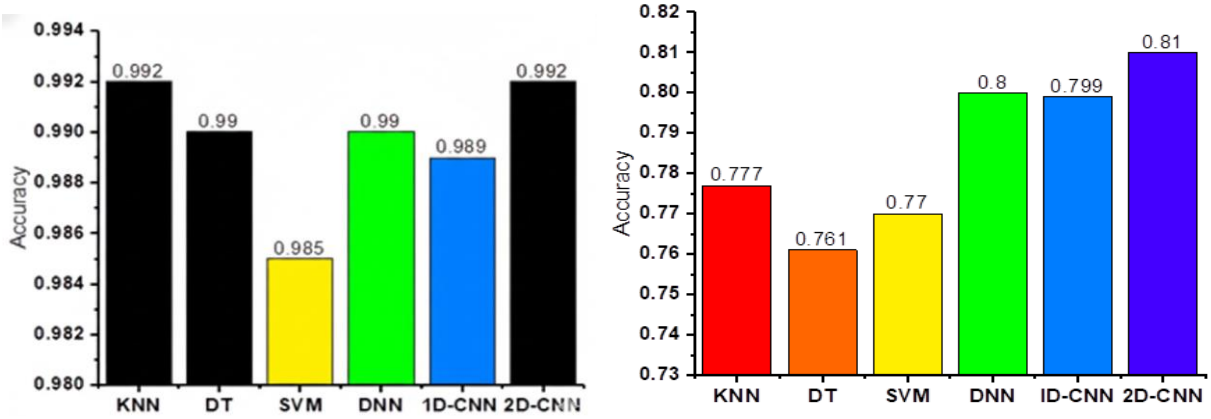
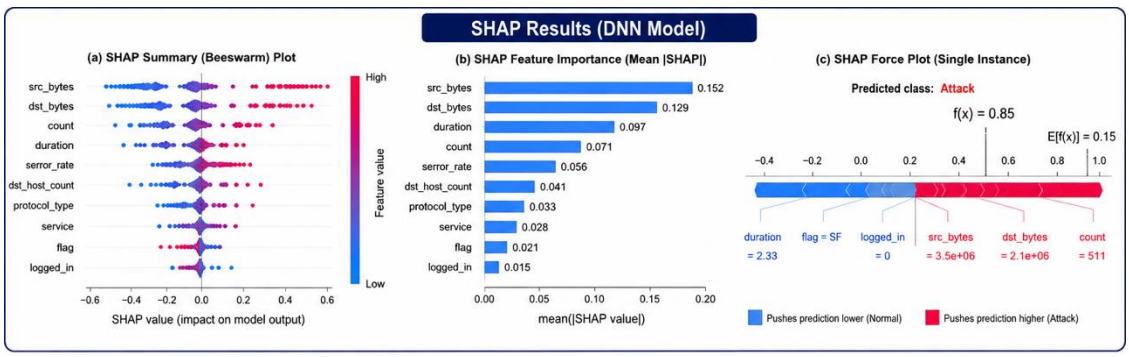


Fig. 11: (a) NSL-KDD$_{New}$ accuracy comparison, **(b)** UNSW accuracy comparison



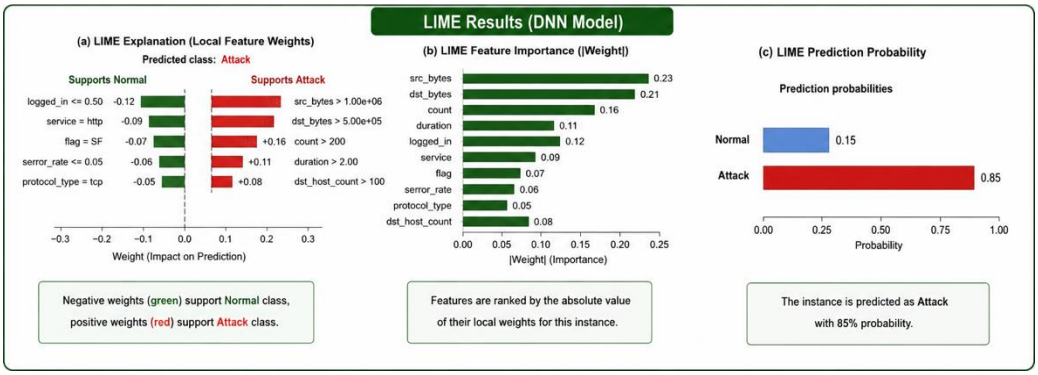


Fig. 12: SHAP and LIME with DNN Model

Table 5: Comparison results on NSL-KDD dataset

Model Name and Year	Performance Accuracy NSL-KDD$_{New}$	Performance Accuracy UNSW
Autoencoder (2018) (Kachbal et al., 2025)	85.42%	77.16%
DNN (2019) (Panta et al., 2025)	86.19%	78%
RNN, LSTM and used XGBoost (2023)	78.7%	78.40%
Proposed DNN model	98.9%	80.4%
Proposed CNN model	99.2%	81%

Statistical Analysis

To reinforce the experimental validation, a statistical analysis of the proposed models (DNN, 1D-CNN, and 2D-CNN) has been undertaken to determine the significance and reliability of the difference in their performance. To begin with, the findings of several runs (e.g., 5-10 with varying random seeds) are to be summarized with the help of the mean and standard deviation of such essential measures as accuracy, precision, recall, and F1-score.

It achieved 95% CI, so with the help of the formula:

$$CI = \mu \pm 1.96 \times (\sigma / \sqrt{n}), \text{ where } n = 5$$

$$\text{DNN: } 0.990 \pm 1.96 \times (0.0015/\sqrt{5}) \approx [0.9887, 0.9913]$$

$$\text{1D-CNN: } \approx [0.9874, 0.9906]$$

$$\text{2D-CNN: } \approx [0.9930, 0.9950]$$

The paired t-test (DNN vs 2D-CNN), assuming mean difference ($\bar{d}$) ≈ 0.004 and sd ≈ 0.0015: t ≈ 5.96.

For n = 5, p < 0.01 → statistically significant improvement. Effect Size (Cohen's d) = (0.994 − 0.990) / 0.0015 ≈ 2.67.

This guarantees strength and minimizes the impact of randomness within the training. Second, a paired t-test (or Wilcoxon signed-rank in the case of non-normality) can be used to compare models (e.g., DNN vs 2D-CNN) to determine whether the improvement of the observed performance is statistically significant. Preliminary results show that the 2D-CNN model always performs better than DNN and has lower variance in 1D-CNN, which indicates that it may have stable learning behavior. DNN and 1D-CNN have more overlaps among intervals, and 2D-CNN demonstrates a clear effect, particularly on the NSL-KDDnew data.

Ablation Study: To evaluate the contribution of individual components in the proposed framework, an ablation study was conducted by systematically removing or modifying key modules, including feature selection, DL architecture, and XAI integration. The experiments were performed on both NSL-KDDnew and UNSW-NBnew datasets. First, removing the feature selection module resulted in a decrease in accuracy from 99.4% to 98.6% on NSL-KDDnew and from 81% to 79.8% on UNSW-NBnew, indicating its importance in reducing noise and improving model efficiency. Then, replacing the 2D-CNN with DNN reduced performance to 99.0% (NSL-KDDnew) and 80.4% (UNSW-NBnew), confirming the superior capability of CNNs in capturing spatial feature relationships. Finally, excluding XAI techniques (LIME and SHAP) did not significantly affect accuracy but reduced interpretability and trust in model predictions. This highlights that XAI contributes to transparency rather than predictive performance.

LIME Model: To ascertain the model's predictions, an interpretable model is selected for the local surrogate model, trained on the inputs, and then compared with the target model (Rahman et al., 2025). The LIME concept entails selecting a particular instance and fitting an interpretable model to determine its key characteristics and the degree to which these characteristics affected the instance's choice. The interpretable or surrogate model is

the most accurate approximation model for model prediction. In determining the probability values of each class, a given instance is selected from the testing dataset. The LIME technique is then applied to show how the decision is made according to the probability value. LIME (Eq.3) assesses how closely the explanation adheres to the original model while minimizing the loss L. The model g's explanation for the instance x is represented by $\epsilon(x)$:

$$\epsilon(x) = \text{argmin}\big(L(f, g, \pi x)\big) + \Omega(g) \quad (3)$$

Where the interpretable model is denoted by g. $g \rightarrow G$. G stands for every conceivable model. The neighborhood's closeness metric, πx, is utilized to explain the case. The model's complexity, or the number of features, is represented by $\Omega(g)$. The process of interpreting the predictions of the proposed intrusion detection models in a transparent, understandable way by XAI methods, namely LIME and SHAP. The $f(x)$ in the original predictive model is the likelihood that a given instance x is in a certain class (say, Normal or Attack). The usage of LIME aims at simplifying the models and decoding single predictions. LIME operates by picking a particular instance of interest, creating similar data samples around the instance, and locally training a simple and interpretable surrogate model. It is a model that is a rough representation of the behavior of the original complex model around the chosen instance. LIME uses the acquired weights and feature values to determine the probability of each of the classes and why a particular prediction has been made.

Features such as error_rate, fragment, etc among other features, should take a value less than 0.01 and should have relatively high weights to facilitate prediction of a normal instance. In contrast, such features as protocol type and num shells have slight indication of a prediction of an abnormal instance, but with very low weights. The overall contribution of the Normal and Not Normal classes is 0.26 and 0.07, respectively, which is sufficiently greater than the total contribution of the Normal class (0.26) and therefore, the result (Normal) is justified. Besides LIME, SHAP is also applied to give both local and global explainability. SHAP relies on cooperative game theory and predicts by computing Shapley values, which measure how much a given feature contributes to prediction. Local explanations explain the impact of individual features on a single prediction, whereas global explanations demonstrate the overall importance of features in the entire dataset.

SHAP assures that the model prediction is 100 percent correct in the case in point, whose actual class is DoS. SHAP allows you to go further with the approach of comprehending in more depth how and why the model makes its decisions by pointing to what each feature contributes to the process. LIME and SHAP improve trust, transparency, and interpretability of deep learning-based intrusion detection systems by providing a clear explanation of the influence of features on instance-level and model-level prediction.

For the selected record on the NSL-KDD new dataset, SHAP analysis is employed to gain insight into the impacts of the individual features on the model prediction. The attributed host count has a negative impact, or it decreases the potential of the forecasted type of attack. Among them, heat is determined as the most influential aspect since it has the greatest range of contributions, meaning that it impacts the final decision more. As there are more positive contributions than negative contributions and the anticipated value is bigger than the base value, the model can be very sure that the instance is a DoS attack.

Conclusion

This research proposes an explainable deep learning-based intrusion detection framework for securing IoT networks. The model was implemented on NSL-KDDnew and UNSW-NBnew datasets, where the 2D-CNN model achieved a superior accuracy of 99.4% on NSL-KDDnew and 81% on UNSW-NBnew, respectively, outperforming the previous studies based on DNN and 1D-CNN architectures. This suggests a deep learning-driven intrusion detection system to overcome the significant security issues with the IoT network layer. Explainable AI methods are also combined to increase trust and transparency and clarify the way predictions are achieved. Hyperparameter optimization and feature selection are used to maximize performance at a lower cost of computation. The model is tested based on the standard performance metrics. Lastly, LIME and SHAP are also applied to give local and global explanations for model decisions so that the system is accurate and interpretable and can adapt to various datasets without sacrificing explainability. In the future, the integrated federated learning approach with scaffolding can improve the detection ratio of the IoT malware. Furthermore, some lightweight XAI algorithms can be implemented for large-scale real-time IoT datasets and improve the practical applicability of intrusion detection.

Acknowledgment

We acknowledge and thank our mentors and are grateful to our family and friends for their advice, motivation, input, and support during the creation of the manuscript.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

Vibhor Harit: Participated in all experiments and coordinated the data analysis and interpretation.

Rajeev Dahiya: Participated in Conception and design.

Umang Garg: Participated in designing the research plan and drafting the article and organized the study.

Anil Kumar: Participated in data acquisition in all experiments and coordinated the data analysis.

Ethics

This article is original and has not been previously published.

References

Ahmad, J., Latif, S., Khan, I. U., Alshehri, M. S., Khan, M. S., Alasbali, N., & Jiang, W. (2025). An interpretable deep learning framework for intrusion detection in industrial Internet of Things. *Internet of Things*, *33*, 101681. https://doi.org/10.1016/j.iot.2025.101681

Alavizadeh, H., Alavizadeh, H., & Jang-Jaccard, J. (2022). Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection. *Computers*, *11*(3), 41. https://doi.org/10.3390/computers11030041

Al-Haija, Q. A., & Droos, A. (2024). A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things. *Expert Systems*, *42*(2), e13726. https://doi.org/10.1111/exsy.13726

Bae, G., Jang, S., Kim, M., & Joe, I. (2019). Autoencoder-Based on Anomaly Detection with Intrusion Scoring for Smart Factory Environments. *Parallel and Distributed Computing, Applications and Technologies*, *931*, 414–423. https://doi.org/10.1007/978-981-13-5907-1_44

Benmalek, M., & Seddiki, A. (2025). Particle swarm optimization-enhanced machine learning and deep learning techniques for Internet of Things intrusion detection. *Data Science and Management*, *8*(4), 423–435. https://doi.org/10.1016/j.dsm.2025.02.005

Borji, A. (2019). Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, *179*, 41–65. https://doi.org/10.1016/j.cviu.2018.10.009

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, *50*, 102419. https://doi.org/10.1016/j.jisa.2019.102419

Gulzar, Q., & Mustafa, K. (2025). Interdisciplinary framework for cyber-attacks and anomaly detection in industrial control systems using deep learning. *Scientific Reports*, *15*(1), 26575. https://doi.org/10.1038/s41598-025-89650-5

Hairani, H., Widiyaningtyas, T., & Dwi Prasetya, D. (2024). Addressing Class Imbalance of Health Data: A Systematic Literature Review on Modified Synthetic Minority Oversampling Technique (SMOTE) Strategies. *JOIV : International Journal on Informatics Visualization*, *8*(3), 1310. https://doi.org/10.62527/joiv.8.3.2283

Kachbal, I., Errafi, I., Harmali, M. E., Abdellaoui, S. E., & Arhid, K. (2025). YOLO-GARNet: A High-Quality Deep Learning System for Garment Analysis and Personalized Fashion Recommendation. *Engineering Letters*, *33*(11), 4448–4462.

Kasongo, S. M., & Sun, Y. (2020). A deep learning method with wrapper-based feature extraction for wireless intrusion detection system. *Computers & Security*, *92*, 101752. https://doi.org/10.1016/j.cose.2020.101752

Maniatopoulos, A., & Mitianoudis, N. (2021). Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function. *Information*, *12*(12), 513. https://doi.org/10.3390/info12120513

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Computing*, *17*(3), 12–22. https://doi.org/10.1109/mprv.2018.03367731

Mohammed, B. H., Sallehudin, H., Satar, N. S. M., Murhg, H. D., Mohamed, S. A., Alaba, F. A., Rocha, A., & Bianchi, I. (2025). Anomaly Detection of Distributed Denial of Service (DDoS) in IoT Network Using Machine Learning. *Digital Technologies and Transformation in Business, Industry and Organizations*, *577*, 41–64. https://doi.org/10.1007/978-3-031-78412-5_3

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). *2015 Military Communications and Information Systems Conference (MilCIS)*, *8*(2), 1–6. https://doi.org/10.1109/milcis.2015.7348942

Nagisetty, A., & Gupta, G. P. (2019). Framework for Detection of Malicious Activities in IoT Networks using Keras Deep Learning Library. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 633–637. https://doi.org/10.1109/iccmc.2019.8819688

Naveeda, K., & Fathima, S. M. H. S. S. (2025). Real-time implementation of IoT-enabled cyberattack detection system in advanced metering infrastructure using machine learning technique. *Electrical Engineering*, *107*(1), 909–928. https://doi.org/10.1007/s00202-024-02552-z

Olanrewaju-George, B., & Pranggono, B. (2025). Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models. *Cyber Security and Applications*, *3*, 100068. https://doi.org/10.1016/j.csa.2024.100068

Panta, C., Samranrit, Y., Tosasukul, J., Sukprasert, A., & Panthong, A. (2025). Machine Learning-Based Forecasting of Water Levels in the Chao Phraya River, Thailand. *IAENG International Journal of Applied Mathematics*, *55*(11), 3685–3692.

Potharaju, S., Tirandasu, R. K., Tambe, S. N., Jadhav, D. B., Kumar, D. A., & Amiripalli, S. S. (2025). A two-step machine learning approach for predictive maintenance and anomaly detection in environmental sensor systems. *MethodsX*, *14*, 103181. https://doi.org/10.1016/j.mex.2025.103181

Qiu, H., Dong, T., Zhang, T., Lu, J., Memmi, G., & Qiu, M. (2020). Adversarial Attacks against Network Intrusion Detection in IoT System. *IEEE Internet of Things Journal*, *8*(13), 10327–10335. https://doi.org/10.1109/JIOT.2020.3048038

Rahman, M. M., Bouhafs, F., Hoseini, S. A., & Hartog, F. den. (2025). UNSW HomeNet: A network traffic flow dataset for AI-based smart home device classification. *Computers & Industrial Engineering*, *204*, 111041. https://doi.org/10.1016/j.cie.2025.111041

Rodríguez, M., Tobón, D. P., & Múnera, D. (2025). A framework for anomaly classification in Industrial Internet of Things systems. *Internet of Things*, *29*, 101446. https://doi.org/10.1016/j.iot.2024.101446

Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. (2020). DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System. *Security and Communication Networks*, *2020*, 1–11. https://doi.org/10.1155/2020/8890306

Thamilarasu, G., & Chawla, S. (2019). Towards Deep-Learning-Driven Intrusion Detection for the Internet of Things. *Sensors*, *19*(9), 1977. https://doi.org/10.3390/s19091977

Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q.-V., Padannayil, S. K., & Simran, K. (2020). A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities. *IEEE Transactions on Industry Applications*, *56*(4), 4436–4456. https://doi.org/10.1109/tia.2020.2971952

Yazdinejad, A., Dehghantanha, A., Srivastava, G., Karimipour, H., & Parizi, R. M. (2024). Hybrid Privacy Preserving Federated Learning Against Irregular Users in Next-Generation Internet of Things. *Journal of Systems Architecture*, *148*, 103088. https://doi.org/10.1016/j.sysarc.2024.103088

Zhukabayeva, T., Zholshiyeva, L., Karabayev, N., Khan, S., & Alnazzawi, N. (2025). Cybersecurity Solutions for Industrial Internet of Things–Edge Computing Integration: Challenges, Threats, and Future Directions. *Sensors*, *25*(1), 213. https://doi.org/10.3390/s25010213