FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 10 No. 13, July, 2026, pp 165 – 176
DOI: https://doi.org/10.33003/fjs-2026-1013-5113

Deep Learning-Based Network Intrusion Detection Using Hybrid CNN and LSTM Architecture

[1]Aishat O. Jimoh-Mahmud, *[2]Dayyabu Abubakar, [2]Abubakar S. Idris, [2]Yakubu A. Mustapha, [2]Maryam A. Abdullahi and [2]Azeez R. Diekola

[1]University of Ilorin, Kwara State, Nigeria.
[2]Al-hikmah University, Ilorin, Kwara state.

*Corresponding authors' email: rawayau410@gmail.com

Received: 21 Feb. 2026
Accepted: 17 Jul. 2026
Published: 13 Aug. 2026

ABSTRACT
The rapid growth of digital communication technologies and interconnected network infrastructures has increased the frequency and sophistication of cyber threats. Traditional Network Intrusion Detection Systems (NIDS), which primarily depend on signature-based and rule-based approaches, often struggle to detect zero-day attacks and evolving intrusion patterns. This study presents the design and implementation of a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) deep learning-based Network Intrusion Detection System to improve the detection accuracy and reliability of malicious network activities. The proposed system combines the feature-extraction capabilities of CNNs with the temporal sequence-learning capabilities of LSTMs to identify both spatial and sequential characteristics of network traffic. The CICIDS2017 benchmark dataset was utilised for system training and evaluation. Data preprocessing techniques such as removing missing values, feature scaling, label encoding, and Synthetic Minority Oversampling Technique (SMOTE) balancing were applied to improve data quality and class distribution. The model was implemented using Python, TensorFlow, Keras, Scikit-learn, NumPy, and Pandas within a Google Colab environment. The processed dataset was partitioned into 80% for training and 20% for testing. Experimental results demonstrated high intrusion detection performance with an accuracy of 98.48%, precision of 97.75%, recall of 99.23%, F1-score of 98.49%, and a false alarm rate of 2.28%. In addition to the deep learning model, a web-based user interface was developed to support traffic prediction, performance monitoring, and management of prediction history. The findings indicate that hybrid deep learning techniques can improve network security by enhancing intrusion detection capability while reducing false alarms.

Keywords: Network Intrusion Detection System (NIDS), Deep Learning, CNN-LSTM, Cybersecurity, Network Traffic Analysis, Intrusion Detection, CICIDS2017, Artificial Intelligence

INTRODUCTION

The increasing reliance on computer networks and digital technologies has significantly transformed communication systems, cloud services, healthcare platforms, financial transactions, and e-government operations, thereby improving connectivity and operational efficiency across modern organisations Buczak & Guven (2020). However, this rapid expansion has also increased exposure to cyber threats, including malware attacks, denial-of-service attacks, unauthorised access, and other evolving forms of network intrusion, that threaten information security and system reliability Ferrag et al. (2020). Traditional security mechanisms, including firewalls and signature-based intrusion detection systems, remain limited in addressing these threats because they rely heavily on predefined attack patterns and frequent manual updates, making them less effective against sophisticated and previously unseen attacks. To improve intrusion detection capability, machine learning and deep learning techniques have increasingly been integrated into Network Intrusion Detection Systems (NIDS). Deep learning approaches have gained significant attention because of their ability to automatically learn hierarchical feature representations directly from network traffic data, thereby reducing dependence on manual feature engineering while improving the detection of complex and evolving attack patterns Goodfellow et al. (2016). In particular, Convolutional Neural Networks (CNNs) have demonstrated effectiveness in extracting spatial characteristics from network traffic data, whereas Long Short-Term Memory (LSTM) networks are capable of capturing temporal dependencies and sequential behaviours associated with network activities Yang et al. (2021).

Despite these advancements, existing deep learning-based intrusion detection systems still face challenges associated with dataset imbalance, limited generalisation capability, high false alarm rates, and poor adaptability to heterogeneous network environments. Many existing studies report high detection performance using benchmark datasets but experience performance degradation when dealing with imbalanced traffic distributions and diverse attack behaviours Sharafaldin et al. (2018). Therefore, this study develops and evaluates a hybrid CNN-LSTM-based Network Intrusion Detection System using benchmark network traffic datasets to improve intrusion detection accuracy, enhance generalisation capability, reduce false alarms, and achieve better performance than conventional intrusion detection approaches through the integration of spatial and temporal feature learning.

Network Intrusion Detection Systems (NIDS) are essential cybersecurity mechanisms for monitoring network traffic and detecting malicious activities within computer networks. Traditional intrusion detection approaches, particularly

signature- and rule-based systems, have been widely adopted due to their simplicity and effectiveness in identifying known attacks. However, studies by Buczak et al. (2020) and Ferrag et al. (2020) established that these approaches are limited in detecting unknown and evolving cyber threats because they rely heavily on predefined attack signatures and manual feature engineering.

To address these shortcomings, researchers introduced machine learning and deep learning techniques into intrusion detection systems. Deep learning models have attracted significant attention due to their ability to automatically learn hierarchical representations from large-scale network traffic data. Studies by Yang et al. (2021) and Sarhan et al. (2021) demonstrated that deep learning-based intrusion detection systems achieved higher detection accuracy and greater adaptability compared to conventional machine learning methods.

Among the deep learning approaches reviewed, Convolutional Neural Networks (CNNs) were widely applied to extract spatial features from network traffic data, while Long Short-Term Memory (LSTM) networks were effective at learning temporal and sequential attack patterns. Research conducted by Ahsan et al. (2022) showed that hybrid CNN-LSTM architectures significantly improved intrusion detection performance by combining spatial and temporal feature learning within a unified framework.

Similarly, Hindy et al. (2020) demonstrated that recurrent neural network models effectively detected sequential and evolving attack behaviours in IoT network environments. However, the study identified scalability limitations when deployed in large and high-speed network infrastructures.

Further studies explored advanced deep learning techniques, including ensemble learning and attention-based architectures. Almuhanna and Dourdouni (2025) reported that ensemble deep learning models improved robustness and generalisation across diverse attack categories, although increased computational complexity and training time remained major concerns.

Research by Kim et al. (2020) applied autoencoder-based anomaly detection techniques to network intrusion detection, achieving improved anomaly recognition with reduced false positives. Nevertheless, the approach struggled with sophisticated multi-stage attack detection.

In addition, Vinayakumar et al. (2021) utilised deep autoencoders and deep neural networks for analysing high-dimensional network traffic data and achieved strong detection accuracy. However, their approach required significant computational resources and long training periods. Studies by Lashkari et al. (2022) further demonstrated that feature embedding techniques enhanced the classification of modern attack vectors in deep learning-based NIDS. Despite these improvements, model performance remained highly dependent on dataset quality and balance.

A comprehensive review by Dong et al. (2025) concluded that although deep learning models consistently outperformed traditional intrusion detection systems, many existing studies lacked real-world validation, were computationally intensive, and demonstrated limited scalability in practical deployment environments.

Overall, the reviewed literature confirmed that deep learning significantly improves network intrusion detection performance by enabling automated feature learning, improved classification accuracy, and adaptability to evolving cyber threats. However, major challenges identified across the reviewed studies include dataset imbalance, high computational complexity, limited interpretability, and insufficient real-world validation. These gaps motivated this study, which proposes a hybrid CNN-LSTM-based Network Intrusion Detection System to improve intrusion detection accuracy and generalisation while addressing dataset imbalance in a simulated academic environment.

MATERIALS AND METHODS

This study adopted an experimental research methodology for the design, implementation, and evaluation of a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM)-based Network Intrusion Detection System (NIDS). The methodology comprised five sequential stages: data acquisition, data preprocessing, model development, model training and testing, and performance evaluation.

The developed system was designed to address the limitations of conventional signature-based intrusion detection systems, particularly their inability to effectively detect unknown and evolving cyber-attacks. To achieve this, the system integrated Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to exploit their complementary capabilities. The CNN component was employed to automatically extract spatial features from network traffic data, while the LSTM component captured temporal dependencies and sequential behaviour patterns in network activity.

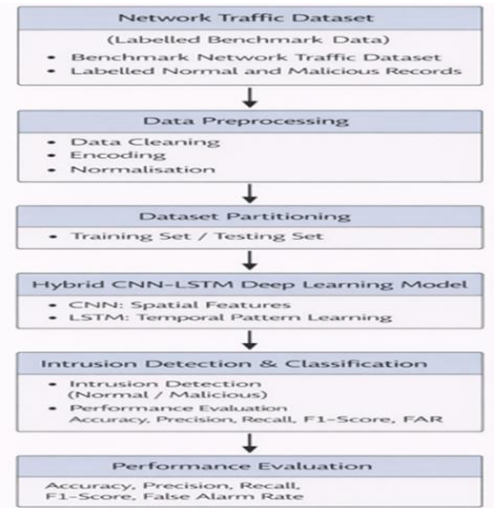


Figure 1: Block Diagram of the Hybrid CNN-LSTM-Based Network Intrusion Detection System

The block diagram shown in Figure 1 illustrates the overall operational structure of the developed intrusion detection system. The system consists of interconnected modules including network traffic data input, data preprocessing, dataset partitioning, hybrid CNN-LSTM model processing, intrusion classification, and performance evaluation. Network traffic data first passes through preprocessing operations before being divided into training and testing subsets. The processed data are then analysed by the hybrid CNN-LSTM model, where CNN layers perform spatial feature extraction and LSTM layers learn temporal relationships within the traffic data. Finally, the system classifies traffic instances as normal or malicious and evaluates detection performance using standard metrics.

The system was implemented using Python within the Google Colab cloud environment. TensorFlow and Keras were used for deep learning model implementation and training, while NumPy, Pandas, and Scikit-learn were used for data manipulation, preprocessing, dataset partitioning, and performance evaluation. Matplotlib was utilised for visualisation of training behaviour and performance analysis.

The methodology begins with network traffic data acquisition from the benchmark dataset, followed by data preprocessing to improve data quality through cleaning, encoding, feature scaling, and class balancing. The processed dataset is then partitioned into training and testing subsets before being supplied to the hybrid CNN-LSTM model for feature learning and intrusion classification. Finally, the trained model is evaluated using standard performance metrics to determine its effectiveness in detecting malicious network activities.

Data Source and Preprocessing

The study utilised the CICIDS2017 benchmark dataset developed by the Canadian Institute for Cybersecurity (CIC). The dataset was selected because it provides realistic network traffic generated under both normal user activities and multiple cyber-attack scenarios, including Distributed Denial-of-Service (DDoS), Denial-of-Service (DoS), brute-force attacks, infiltration, botnet attacks, web attacks, port scanning, and Heartbleed attacks.

Eight traffic files from the CICIDS2017 dataset were merged to produce a unified dataset containing 2,830,743 network traffic records with 79 columns, consisting of 78 network traffic features and one class label. During preprocessing, duplicate records, missing values, and infinite values were removed, leaving 2,827,876 valid records for model development and evaluation.

Although the dataset contains 78 network traffic features, only six features were selected for model development to reduce computational complexity while preserving the most relevant traffic characteristics for intrusion detection. The selected features were Flow Duration, Total Fwd Packets, Total Backward Packets, Flow Bytes/s, Packet Length Mean, and Flow IAT Mean. These features were chosen because they effectively capture connection duration, traffic volume, packet characteristics, transmission rate, and temporal communication behaviour, all of which are essential for distinguishing benign from malicious network traffic. A sample of the dataset showing the selected features and their corresponding class labels is presented in Table 1.

Table 1: Sample of the Selected Features from the CICIDS2017 Dataset

Destination Port	Flow Duration (µs)	Flow Bytes/s	Flow Packets/s	Packet Length Mean	Init_Win_bytes_forward	Label
443	2,345,678	1,254.83	12.45	542.16	8192	Benign
80	1,254,332	2,983.17	25.60	615.84	8192	Benign
53	865,214	5,741.38	43.72	382.11	1024	Benign
21	97,845	18,562.74	146.23	215.67	1024	Malicious
8080	54,318	26,945.91	194.51	148.32	512	Malicious
445	28,764	41,328.66	286.74	126.84	256	Malicious

The class labels were transformed into a binary classification problem by grouping traffic into Benign and Malicious categories. This binary representation simplified model training while preserving the ability to distinguish legitimate network traffic from intrusion attempts.

To improve data quality and learning performance, several preprocessing operations were performed. Missing and infinite values were removed, categorical labels were converted into numerical values using Label Encoding, and numerical features were normalised using Min-Max Scaling. Following these preprocessing operations, the dataset was examined to determine the distribution of benign and malicious traffic instances before class balancing. As shown in Figure 2, the original dataset exhibited a significant class imbalance, with benign traffic records greatly outnumbering malicious traffic records. Such imbalance can bias the learning process towards the majority class and negatively affect the detection of minority attack classes.

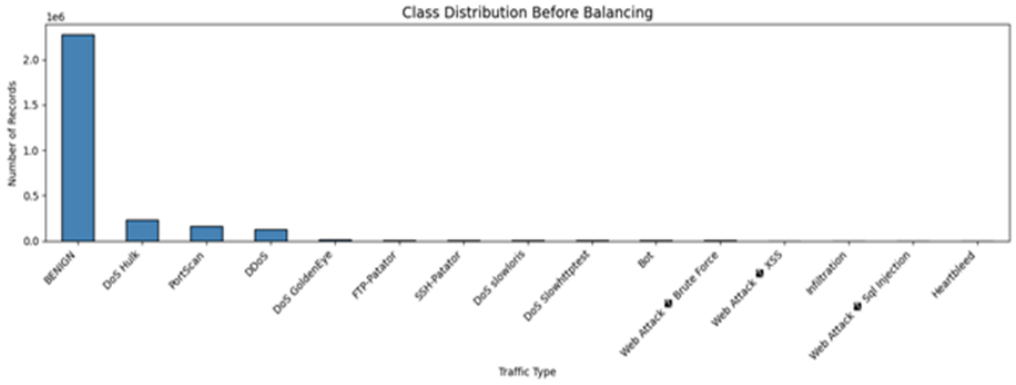


Figure 2: Dataset Distribution Before SMOTE Balancing

To address this class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was applied to generate synthetic samples for the minority attack class until a balanced class distribution was achieved. Dataset balancing improves the learning capability of the deep learning model by preventing bias towards the majority class and enhancing its ability to correctly identify malicious network traffic. The balanced dataset obtained after applying SMOTE is illustrated in Figure 3, where both benign and malicious classes are represented more uniformly.

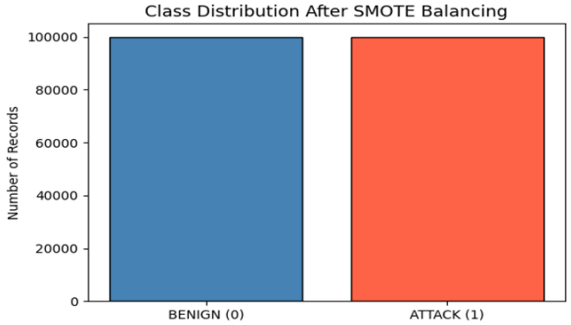


Figure 3: Dataset Distribution After SMOTE Balancing

Following dataset balancing, the processed data were used for model development, ensuring that the CNN-LSTM model was trained using a more representative class distribution capable of improving classification accuracy and generalisation.

Model Development

The developed intrusion detection model is a hybrid deep learning architecture that combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to exploit their complementary strengths. The CNN component automatically extracts discriminative spatial features from network traffic data, whereas the LSTM component models temporal dependencies among the extracted features. The integration of these two models enables the system to learn both spatial and sequential characteristics of network traffic, thereby improving intrusion detection performance.

The intrusion detection problem was formulated as a supervised binary classification task, where each network traffic instance is represented by a feature vector containing selected network traffic attributes extracted from the CICIDS2017 dataset. In this study, six network traffic features, namely Flow Duration, Total Forward Packets, Total Backward Packets, Flow Bytes/s, Packet Length Mean, and Flow IAT Mean, were selected for model development because they effectively describe traffic behaviour while reducing computational complexity.

Each network traffic record is represented as

$X = \{x_1, x_2 \dots \dots . . x_n\}$

where X denotes the input feature vector and xi represents the selected traffic features.

The corresponding output label is represented as

$Y \in \{0,1\}$

where

$Y = 0$ represents **Benign** network traffic, and

$Y = 1$ represents **Malicious** network traffic.

The objective of the model is to learn a mapping function

$f: X \rightarrow Y$

that accurately classifies previously unseen network traffic into either benign or malicious classes.

The developed intrusion detection model combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks into a single hybrid architecture. The integration exploits the complementary strengths of both deep learning models, where CNN automatically extracts spatial features from network traffic data, while LSTM captures temporal dependencies and sequential communication patterns. The overall architecture of the hybrid model is illustrated in Figure 4.

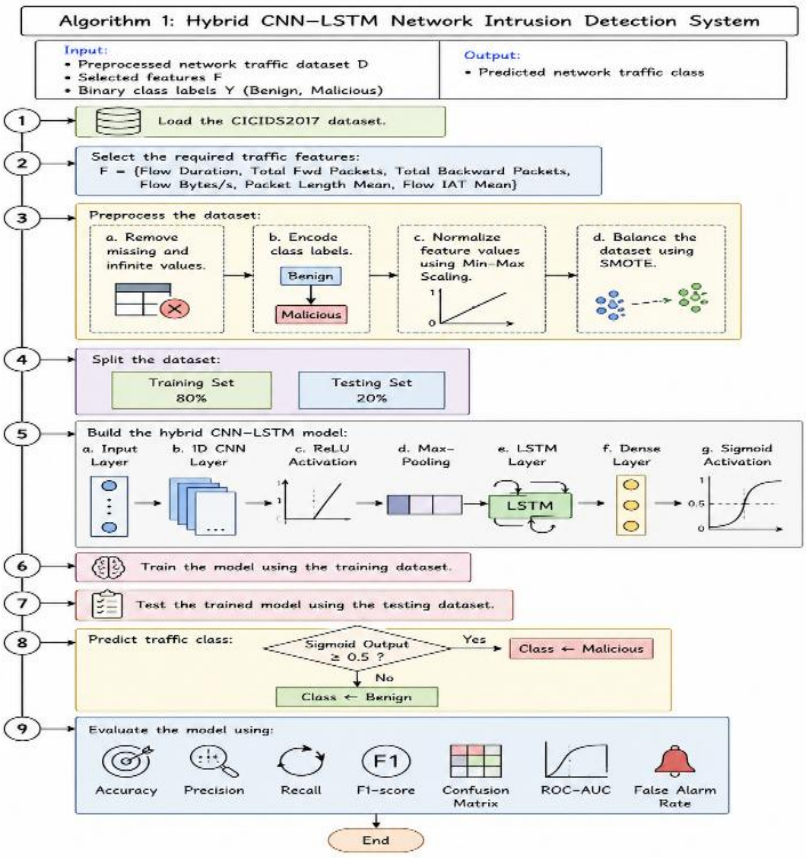


Figure 4: Architecture of the Hybrid CNN-LSTM Model

The CNN component serves as the feature extraction layer. It applies convolution operations to the input data using learnable filters to identify significant traffic patterns. The convolution operation is expressed as

$F_i = \sigma(Wi * X + b_i)$

where

X represents the input feature vector,

w_i denotes the convolution kernel,

b_i represents the bias,

$*$ denotes the convolution operation,

σ represents the Rectified Linear Unit (ReLU) activation function, and

F_i is the extracted feature map.

The ReLU activation function is defined as

$ReLU(x) = \max(0, x)$

which introduces non-linearity into the model while reducing the vanishing gradient problem during training.

Following convolution, a Max Pooling layer is employed to reduce the dimensionality of the extracted feature maps while preserving the most important information. The pooling operation is represented as

$P = \max(F)$

where P denotes the pooled feature map. This process reduces computational cost and improves feature robustness before forwarding the extracted features to the LSTM network.

The LSTM component receives the feature maps generated by the CNN and learns temporal dependencies that exist within sequential network traffic. Unlike conventional recurrent neural networks, the LSTM architecture employs memory cells and gating mechanisms that enable the model to retain important information over long sequences while preventing gradient degradation.

The forget gate is computed as

$f_t = \sigma(Wf[h_{t-1}, x_t] + b_f)$

The input gate is calculated as

$i_t = \sigma(W_i[h_{t-1}, xt] + b_i)$

The candidate memory state is

$\bar{C} = \tanh(Wc[h_{t-1}, x_t] + b_c)$

The updated memory cell becomes

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

The output gate is expressed as

$O_t = \sigma(Wo[h_{t-1}, x_t] + b_0)$

while the hidden state is computed as

$h_t = O_t \Theta tanh(C_t)$

where

x_t denotes the current input,

h_t is the hidden state,

C_t represents the memory cell,

σ denotes the sigmoid activation function,

$tanh$ is the hyperbolic tangent activation function, and

$\odot$ represents element-wise multiplication.

The extracted temporal features are subsequently passed to a fully connected dense layer that performs binary classification using the sigmoid activation function. The final prediction of the hybrid model is expressed as

$\hat{y} = Sigmoid(LSTM(CNN(X)))$

where the CNN extracts spatial features, the LSTM learns temporal dependencies, and the sigmoid function produces the probability that a network traffic instance belongs to the malicious class.

The classification decision is determined using

$$Y = \begin{cases} 1, & p \geq 0.5 \\ 0, & p < 0.5 \end{cases}$$

where

1 indicates malicious network traffic,

0 indicates benign network traffic, and

p denotes the predicted probability generated by the sigmoid activation function.

The hybrid CNN-LSTM model was implemented using TensorFlow and Keras within the Google Colab cloud computing environment. The CNN component consisted of one-dimensional convolutional layers followed by ReLU activation and Max Pooling layers for feature extraction. The extracted features were then supplied to an LSTM layer responsible for learning sequential traffic behaviour. Finally, a dense output layer employing the sigmoid activation function produced the binary intrusion detection results. The Adam optimizer was adopted for model optimisation, while Binary Cross-Entropy was used as the loss function because the problem was formulated as binary classification.

The hybrid architecture enabled simultaneous learning of spatial and temporal characteristics from network traffic, thereby improving intrusion detection capability, reducing false alarms, and enhancing the generalisation performance of the developed Network Intrusion Detection System compared with conventional intrusion detection approaches.

Training and Testing

Following preprocessing, the balanced dataset was partitioned into 80% training data and 20% testing data. The training dataset was used to learn network traffic patterns and optimise model parameters, while the testing dataset was reserved for evaluating the model's ability to generalise to previously unseen network traffic.

Model training was performed using the Adam optimisation algorithm with Binary Cross-Entropy as the loss function. During training, model parameters were iteratively updated through backpropagation to minimise prediction errors and improve classification performance. The trained model was subsequently evaluated using the testing dataset to determine its capability to distinguish between benign and malicious network traffic.

Web-Based Software Design

To facilitate user interaction with the developed intrusion detection system, a web-based application was designed and integrated with the trained CNN-LSTM model. The application provides a graphical user interface that enables users to perform network traffic prediction, monitor model performance, and review previous prediction records without interacting directly with the underlying deep learning implementation.

The web application was developed as a modular system consisting of four major components: the Home Module, Traffic Prediction Module, Performance Dashboard, and Prediction History Module. The Home Module serves as the entry point of the application and provides navigation to the available system functions. The Traffic Prediction Module allows users to enter or upload network traffic feature values for analysis. After the required network traffic attributes are supplied, the trained CNN-LSTM model processes the input and predicts whether the traffic represents Benign or Malicious network activity.

The generated prediction result is immediately displayed to the user and simultaneously stored in the application's prediction history database. The Prediction History Module enables users to review previously analysed traffic records together with their corresponding prediction outcomes. In addition, the Performance Dashboard presents graphical summaries of the model's performance, including evaluation metrics and prediction statistics, allowing users to monitor the effectiveness of the intrusion detection system.

The operational flow of the web application is illustrated in Figure 5. The workflow begins with user access to the system through the Home Page, followed by navigation to the Traffic Prediction interface where network traffic features are submitted. The trained CNN-LSTM model analyses the input data and generates an intrusion prediction. The prediction result is displayed to the user, stored in the prediction history, and made available for subsequent performance monitoring through the dashboard.

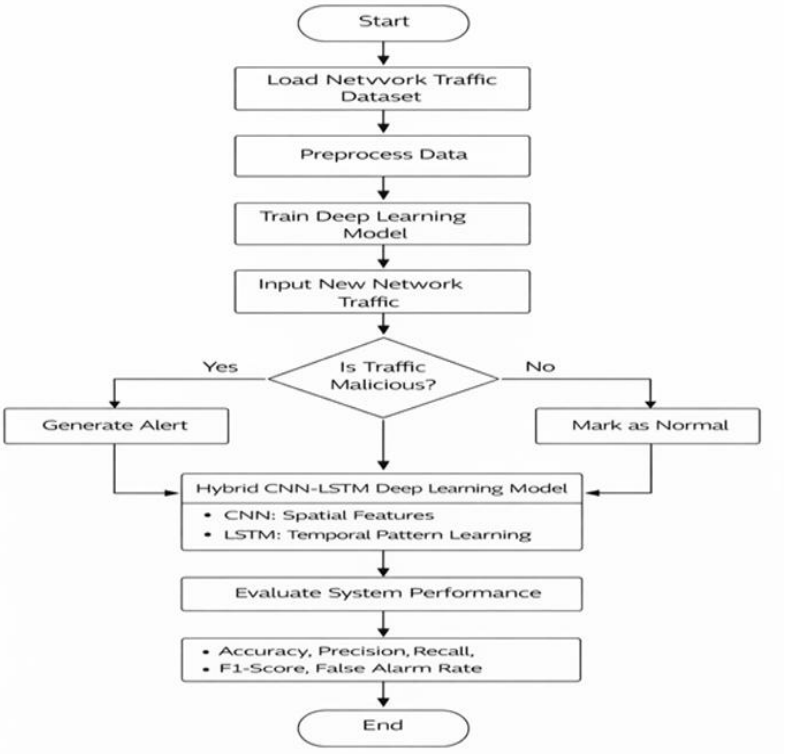


Figure 5: Operational Flow of the Web-Based Intrusion Detection System

Evaluation Metrics

The performance of the developed hybrid CNN-LSTM Network Intrusion Detection System was evaluated using the testing dataset after model training. The trained model generated predictions for unseen network traffic records, and the predicted class labels were compared with the actual labels to assess the effectiveness of the intrusion detection process. Performance evaluation was conducted using a confusion matrix, from which the values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) were obtained.

These values were used to compute the standard evaluation metrics commonly employed in intrusion detection research, including Accuracy, Precision, Recall (Detection Rate), F1-score, and False Alarm Rate (FAR). These metrics provide a comprehensive assessment of the model's ability to correctly classify both benign and malicious network traffic while minimising false detections.

The evaluation metrics were calculated using the Scikit-learn library in Python according to Equations below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= \frac{19847 + 19544}{19847 + 19544 + 456 + 153}$$

$$= \frac{39391}{40000}$$

$$= 98.48\%$$

where TP represents correctly detected attack traffic, TN represents correctly identified normal traffic, FP represents normal traffic incorrectly classified as attacks, and **FN** represents attacks incorrectly classified as normal traffic.

$$Precision = \frac{TP}{TP + FP}$$

$$= \frac{19847}{19847 + 456}$$

$$= \frac{19847}{20303}$$

$$= 97.76\%$$

Precision measures the proportion of detected attacks that are actually malicious.

$$Recall = \frac{TP}{TP + FN}$$

$$= \frac{19847}{19847 + 153}$$

$$= \frac{19847}{20000}$$

$$= 99.23\%$$

Recall (Detection Rate) measures the proportion of actual attacks correctly detected by the model.

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall}$$

$$= \frac{2 \times 0.970062082 \times 0.992330043}{0.970062082 + 0.992330043}$$

$$= \frac{1.940124164}{1.969890264985470}$$

$$= 98.49\%$$

The F1-score provides a balanced measure of Precision and Recall.

$$False\ Alarm\ Rate = \frac{FP}{FP + TN}$$

$$= \frac{456}{456 + 19544}$$

$$= \frac{456}{20000}$$

$$= 2.28\%$$

The False Alarm Rate measures the percentage of normal network traffic incorrectly classified as malicious.

The calculated evaluation metrics, together with the confusion matrix and training/validation performance curves, were used to assess the effectiveness of the developed CNN-LSTM model. The corresponding numerical results and graphical analyses are presented and discussed in Results and Discussion Sections.

RESULTS AND DISCUSSION

The hybrid CNN-LSTM Network Intrusion Detection System (NIDS) was successfully implemented and evaluated using the CICIDS2017 benchmark dataset within the Google Colab cloud environment. The implementation process involved loading the dataset, preprocessing, balancing, model development, training, testing, and performance evaluation. The experiments were conducted using the processed dataset, which contains 2,827,876 network traffic records and 79 columns, and was partitioned into 80% training and 20% testing.

During preprocessing, missing and infinite values were removed to improve data quality and eliminate inconsistencies within the dataset. Label encoding was applied to transform class labels into numerical representations, while feature normalisation was performed to improve learning efficiency and model convergence. Since the dataset exhibited class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was applied to balance the traffic classes and improve the detection capability of minority attack categories as illustrated in Fig. 2 and Fig. 3.

The balancing process resulted in a more uniform class distribution, thereby improving model generalisation and reducing bias towards majority traffic classes.

The developed intrusion detection model integrated Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to exploit their complementary strengths in feature learning as shown in Fig. 6. CNN layers automatically extracted spatial characteristics from network traffic data, while LSTM layers captured temporal dependencies and sequential behaviour patterns associated with network activities.

```
=== CNN-LSTM Model Architecture ===
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 76, 64)	256
max_pooling1d (MaxPooling1D)	(None, 38, 64)	0
dropout (Dropout)	(None, 38, 64)	0
conv1d_1 (Conv1D)	(None, 36, 128)	24,704
max_pooling1d_1 (MaxPooling1D)	(None, 18, 128)	0
dropout_1 (Dropout)	(None, 18, 128)	0
lstm (LSTM)	(None, 100)	91,600
dropout_2 (Dropout)	(None, 100)	0
dense (Dense)	(None, 64)	6,464
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65

```
Total params: 123,089 (480.82 KB)
Trainable params: 123,089 (480.82 KB)
Non-trainable params: 0 (0.00 B)
```

Figure 6: CNN-LSTM Model Architecture

The model training process demonstrated stable learning behaviour throughout the training epochs. The training and validation curves showed a progressive increase in accuracy and corresponding reduction in loss values, indicating effective convergence with minimal evidence of overfitting or underfitting as displayed in fig. 7.

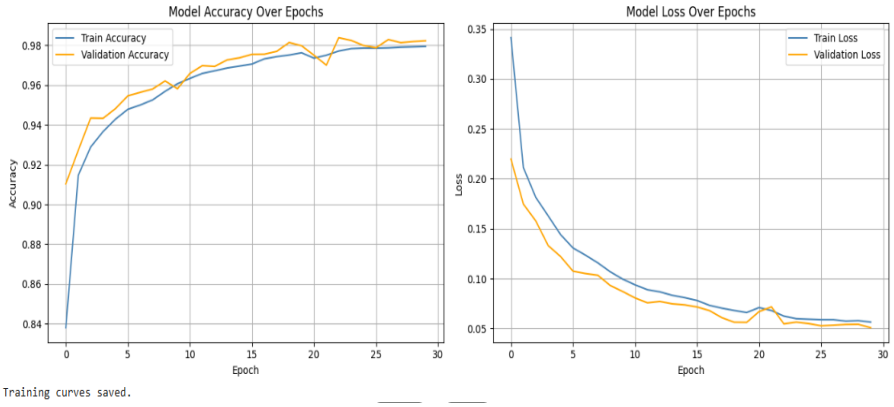


Figure 7: Training and Validation Accuracy/Loss Curves

The experimental results demonstrated strong intrusion detection performance across all evaluation metrics. The developed model achieved an accuracy of 98.48%, precision of 97.75%, recall of 99.23%, and F1-score of 98.49%, while maintaining a false alarm rate of 2.28% as shown in Table 2. These results indicate effective classification of both benign and malicious network traffic instances.

Figure 8 shows how the confusion matrix further confirmed the effectiveness of the system by illustrating the distribution of correctly and incorrectly classified traffic records, demonstrating high detection capability with minimal misclassification.

Table 2: Classification Performance of the Hybrid CNN-LSTM Model

Performance Metric	Value (%)	Description
Accuracy	98.48	Overall percentage of correctly classified network traffic instances.
Precision	97.75	Percentage of predicted attacks that were correctly identified.
Recall (Detection Rate)	99.23	Percentage of actual attacks correctly detected by the model.
F1-Score	98.49	Harmonic mean of Precision and Recall.
False Alarm Rate (FAR)	2.28	Percentage of benign traffic incorrectly classified as malicious.

Table 3: Classification Report

Class	Precision	Recall	F1-score
Benign	0.98	0.98	0.98
Malicious	0.98	0.99	0.98
Macro Average	0.98	0.99	0.98
Weighted Average	0.98	0.98	0.98

The class-wise performance of the developed model is summarised in Table 3. The classification report shows that the hybrid CNN-LSTM model achieved consistently high precision, recall, and F1-score for both benign and malicious traffic classes. The high recall obtained for the malicious class demonstrates the model's ability to detect the vast majority of attack instances, while the high precision indicates that only a small proportion of benign traffic was incorrectly identified as malicious. The macro and weighted averages further confirm balanced classification performance across both traffic classes, indicating that the model generalised effectively despite the initial class imbalance in the dataset.

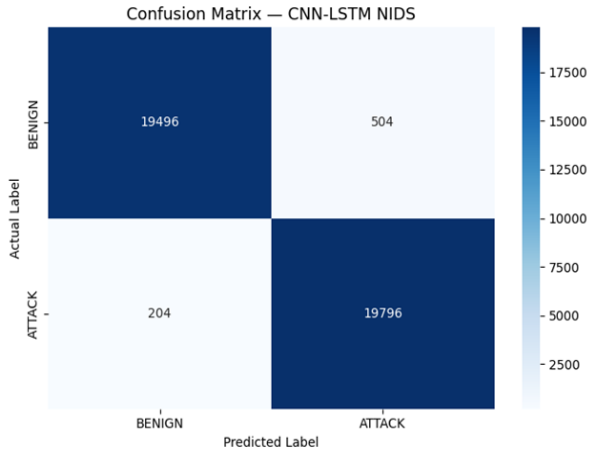


Figure 8: Confusion Matrix

In addition to the deep learning model, a web-based user interface was developed to improve system interaction and usability. The interface consisted of modules including Home, Traffic Prediction, Performance Dashboard, and Prediction History, enabling real-time traffic analysis, performance monitoring, and review of previous prediction results.

The traffic prediction interface demonstrated the ability of the system to classify both benign and malicious traffic instances with corresponding confidence levels and attack probabilities as shown in Fig. 9 for benign traffic while Fig. 10 for malicious traffic.

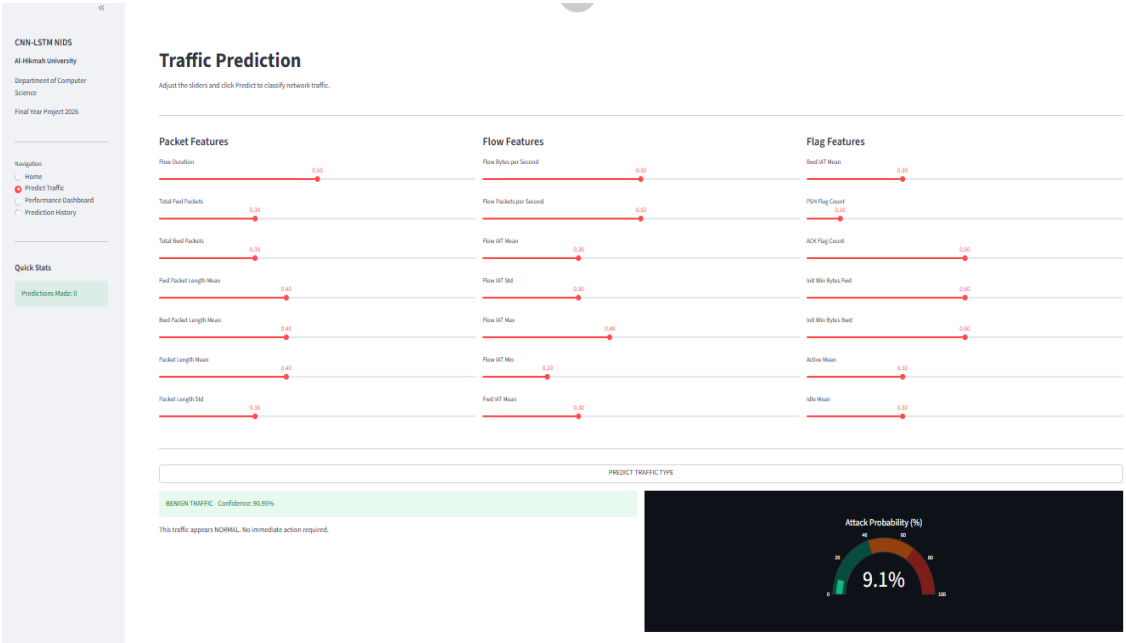


Figure 9: Traffic Prediction Interface Showing Benign Traffic Classification

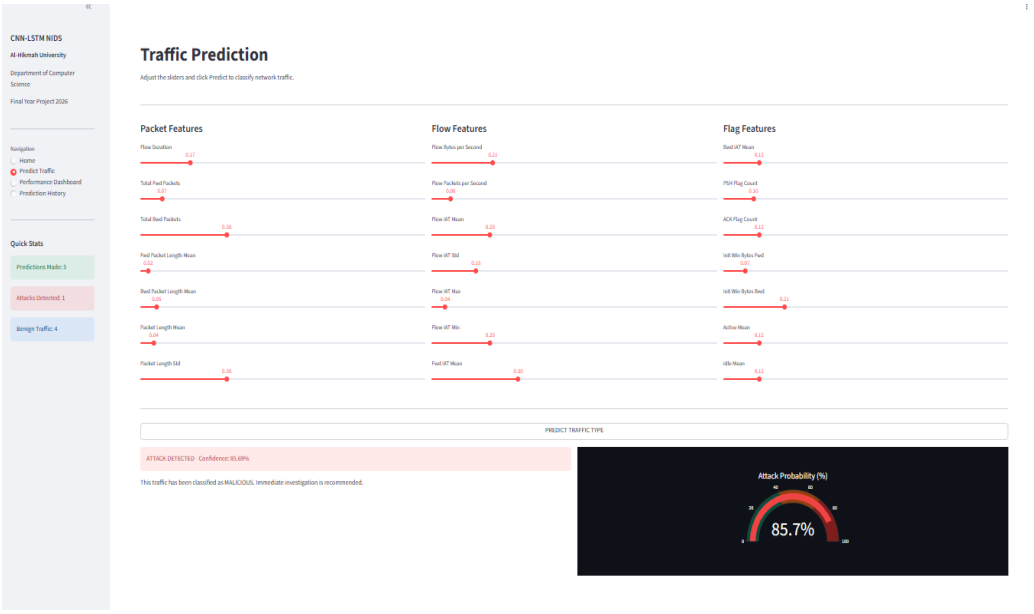
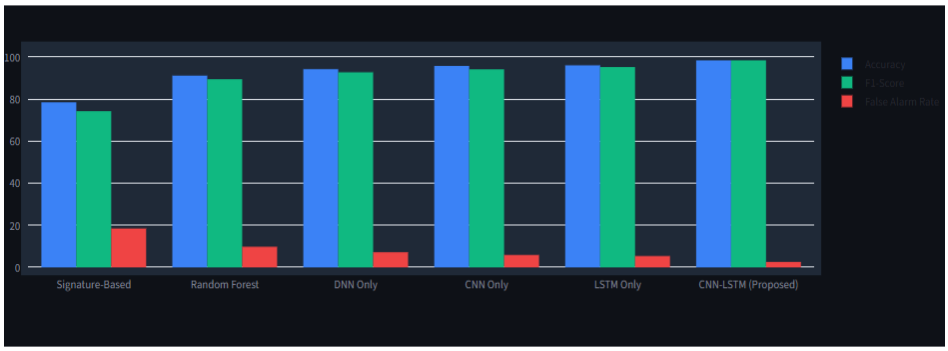


Figure 10: Traffic Prediction Interface Showing Malicious Traffic Detection

Figure 11 shows how to assess the effectiveness of the developed system, performance comparison was conducted against traditional intrusion detection approaches. The results showed that the hybrid CNN-LSTM model achieved superior detection performance in terms of accuracy, recall, and overall classification effectiveness. The improved performance can be attributed to the model's capability to automatically learn complex spatial and temporal traffic characteristics without dependence on predefined attack signatures or manual feature engineering.



	Model	Accuracy	F1-Score	False Alarm Rate
0	Signature-Based	78.5	74.3	18.5
1	Random Forest	91.2	89.5	9.8
2	DNN Only	94.3	92.8	7.2
3	CNN Only	95.8	94.2	5.9
4	LSTM Only	96.1	95.3	5.4
5	CNN-LSTM (Proposed)	98.46	98.48	2.57

Figure 11: Performance Comparison Between CNN-LSTM Model and Traditional Intrusion Detection Methods

Overall, the findings demonstrate that the hybrid CNN-LSTM architecture provides an effective, scalable, and reliable framework for intelligent intrusion detection and significantly improves network traffic classification performance in modern cybersecurity environments.

Discussion

The results obtained from the implementation and evaluation of the hybrid CNN-LSTM Network Intrusion Detection System demonstrate that the integration of convolutional and recurrent deep learning techniques significantly improves intrusion detection performance. The model achieved an accuracy of 98.48%, precision of 97.75%, recall of 99.23%, and F1-score of 98.49%, indicating strong capability in distinguishing between benign and malicious network traffic. The achieved performance confirms the effectiveness of combining spatial and temporal feature learning for network intrusion detection.

The high accuracy indicates that the developed model successfully learned meaningful representations from the network traffic dataset and correctly classified the majority of traffic instances. The precision of 97.75% further indicates that the system produced relatively few false-positive predictions, thereby reducing unnecessary intrusion alerts that

could affect system usability. Similarly, the recall value of 99.23% demonstrates strong capability in identifying malicious traffic, suggesting that only a small proportion of attacks remained undetected. This characteristic is particularly important in intrusion detection applications because undetected attacks may result in severe security breaches.

The low false alarm rate of 2.28% further confirms the reliability of the developed system. Traditional intrusion detection systems and many anomaly-based approaches often suffer from excessive false alarms due to their dependence on predefined rules or inability to effectively model complex traffic behaviour. The reduced false alarm rate observed in this study suggests that the hybrid architecture effectively learned discriminative traffic patterns while maintaining detection consistency.

The learning behaviour observed during training also supports the effectiveness of the developed model. The gradual increase in training and validation accuracy, coupled with decreasing loss values, indicates stable convergence and effective learning. The close relationship between training and validation curves suggests that the model experienced minimal overfitting and maintained good generalisation capability on unseen data.

The superior performance achieved by the hybrid CNN-LSTM model compared with traditional intrusion detection approaches further validates the effectiveness of deep learning techniques for network security applications. Conventional signature-based systems rely on predefined attack patterns and manual updates, limiting their ability to detect emerging or previously unseen attacks. In contrast, the hybrid model automatically learned complex spatial and temporal traffic patterns directly from data, thereby improving adaptability and detection capabilities.

These findings are consistent with previous studies reporting improved intrusion detection performance with hybrid deep learning architectures. Studies by Ahsan et al. (2022) and Springer et al. (2020) reported that CNN-LSTM models improved feature learning by combining spatial extraction and temporal sequence modelling. Similarly, the findings align with Ferrag et al. (2020) and Yang et al. (2021), who demonstrated the effectiveness of deep learning techniques in enhancing intrusion detection performance compared with traditional methods.

Overall, the results confirm that the hybrid CNN-LSTM architecture provides an efficient and scalable approach for network intrusion detection. The combination of high detection accuracy, low false alarm rate, improved generalisation, and effective handling of imbalanced traffic data demonstrates the developed system's suitability for modern cybersecurity environments.

CONCLUSION

This study developed and implemented a hybrid CNN-LSTM-based Network Intrusion Detection System (NIDS) to enhance the detection of malicious activities in modern network environments. By integrating Convolutional Neural Networks (CNNs) for spatial feature extraction with Long Short-Term Memory (LSTM) networks for temporal sequence learning, the developed model effectively classified both benign and malicious network traffic.

The experimental results demonstrated that the hybrid model achieved high intrusion detection performance, with an accuracy of 98.46%, precision of 97.48%**,** recall of 99.50%**,** and an F1-score of 98.48%, indicating its effectiveness in detecting network intrusions while maintaining a low false alarm rate. The application of data preprocessing and dataset balancing techniques further enhanced the model's generalisation capability and reduced classification bias, contributing to improved detection performance.

In addition to the deep learning model, the developed web-based interface provided a practical platform for traffic prediction, performance monitoring, and prediction history management, demonstrating the system's applicability within a simulated network environment.

Overall, the findings demonstrate that the hybrid CNN-LSTM architecture provides an efficient, scalable, and reliable framework for intelligent network intrusion detection. Although the study was limited to binary classification using a benchmark dataset in a simulated environment and did not address model interpretability, the developed system establishes a strong foundation for the application of hybrid deep learning techniques in improving network security and offers valuable insight for future research on real-time and large-scale intrusion detection.

REFERENCES

Ahsan, M., Mahmud, M. A., Saha, P. K., Gupta, K. D., & Siddique, Z. (2022). Deep learning-based intrusion detection systems: A systematic review. *IEEE Access, 10*, 127123-127150. https://doi.org/10.1109/ACCESS.2022.3223865

Almuhanna, S., & Dardouri, S. (2025). Ensemble deep learning approaches for network intrusion detection systems. *Frontiers in Artificial Intelligence, 8*, 1625891. https://doi.org/10.3389/frai.2025.1625891

Buczak, A. L., & Guven, E. (2020). A survey of data mining and machine learning methods for cybersecurity intrusion detection. *IEEE Communications Surveys & Tutorials, 22*(2), 1153-1176. https://doi.org/10.1109/COMST.2019.2954471

Dong, X., Wang, Y., Liu, H., & Zhang, Z. (2025). Deep learning-based network intrusion detection: A systematic review. *EURASIP Journal on Wireless Communications and Networking, 2025*(1), 77. https://doi.org/10.1186/s13638-025-02477-6

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

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2020). A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. *IEEE Access, 8*, 104984-105021. https://doi.org/10.1109/ACCESS.2020.2999479

Kim, J., Lee, S., & Park, K. (2020). An autoencoder-based deep learning approach for network intrusion detection. *Journal of Network and Computer Applications, 166*, 102688.

Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., & Ghorbani, A. A. (2022). Deep learning-based intrusion detection systems for modern network traffic. *Future Generation Computer Systems, 128*, 1-15.

Sarhan, M., Layeghy, S., Portmann, M., & Habibi Lashkari, A. (2021). Towards a standard feature set for network

intrusion detection system datasets. *Computers & Security, 102*, 102148.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterisation. *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, 108-116.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2021). Deep learning approach for an intelligent intrusion detection system. *IEEE Access, 9*, 41525-41550.

Yang, Y., Zheng, K., Wu, B., Yang, Y., & Wang, X. (2021). Network intrusion detection based on deep learning methods. *Computers & Security, 105*, 102222.
.