International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

A Hybrid Deep Learning based Intrusion Detection System

Dr. Ayesha Ameen, [1] Professor, Department of Computer Science and Engineering,

Deccan College of Engineering and Technology, Hyderabad, India

Mahek Rimsha,[2]Mtech Student, Department of Computer Science and Engineering,

Deccan College of Engineering and Technology, Hyderabad, India

ABSTRACT -- The increasing dependence on digital communication networks has made the protection of information systems a significant challenge. As network environments continue to expand, cyber-attacks have become more frequent and diverse, reducing the effectiveness of conventional intrusion detection techniques that depend primarily on predefined attack signatures or manually designed rules. These traditional approaches often struggle to recognise evolving attack patterns and are less effective when dealing with complex network traffic. This project presents a hybrid deep learning framework for network intrusion detection by combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. The CNN component is employed to learn meaningful feature representations from network traffic data, while the LSTM network captures sequential relationships that help distinguish normal behaviour from malicious activities. The integration of these two deep learning models enables the system to analyse both spatial and temporal characteristics of network traffic, leading to more reliable attack classification. The proposed framework is developed and evaluated using the CICIDS2017 dataset, which includes realistic network traffic containing both legitimate and malicious activities. The trained model is deployed through a Flask-based web application that supports user authentication, individual traffic prediction, batch prediction using CSV files, prediction history management, dashboard visualisation, and report generation. SQLite is used to maintain prediction records and application data efficiently. Experimental evaluation indicates that the hybrid CNN–LSTM model achieves strong classification performance across standard evaluation metrics, including accuracy, precision, recall, and F1-score. The combination of deep feature extraction and sequential learning enhances the system's capability to identify malicious network traffic more effectively than conventional machine learning approaches. The developed framework offers a practical and scalable solution for intelligent network intrusion detection and provides a foundation for improving cybersecurity in modern networked environments.

Index Terms— Network Intrusion Detection, Cybersecurity, Deep Learning, Convolutional Neural Network, Long Short-Term Memory, CNN–LSTM, CICIDS2017, Intrusion Detection System, Network Traffic Classification, Anomaly Detection, Binary Classification, Flask, Network Security, Web-Based Monitoring.

1. INTRODUCTION

1.1 Background

The rapid expansion of computer networks has made digital communication essential for organizations, institutions, businesses, and government services. Applications such as cloud computing, online transactions, data exchange, and interconnected systems rely heavily on network infrastructure. However, this growing dependence also increases exposure to cyberattacks, making effective network security and intrusion detection increasingly important.

Conventional **Intrusion Detection Systems (IDS)** generally use predefined signatures, rules, or manually selected features to identify malicious traffic. While these methods can detect known attack patterns, they may be less effective against new or changing threats. The large volume and complexity of modern network traffic further increase the difficulty of accurate and efficient detection.

The development of **Machine Learning (ML)** and **Deep Learning (DL**) has introduced data-driven approaches for identifying abnormal network behaviour. Among deep-learning techniques,

Convolutional Neural Networks (CNNs) can learn useful feature representations, while **Long Short-Term Memory (LSTM)** networks are capable of modelling dependencies in sequential data. Combining these two architectures therefore provides an opportunity to analyse both traffic characteristics and their sequential relationships.

This research proposes a **Hybrid CNN–LSTM Framework for Network Intrusion Detection with Web-Based Monitoring and Analytics**. The **CICIDS2017** dataset is used for model development and evaluation. Network records are pre-processed through cleaning, feature preparation, label encoding, and standardization before being passed through the CNN–LSTM architecture. The resulting representation is classified into two categories, **Normal** and **Attack.**

The framework also includes a **Flask-based web application** that enables individual and CSV-based batch prediction. Prediction results can be stored in SQLite and accessed through history, dashboard, and reporting functions. This provides a practical interface for using the trained model rather than limiting the research to standalone classification.

The proposed system is assessed using **accuracy, precision, recall, F1-score, ROC-AUC, and confusion-matrix analysis**. Functional testing is also used to examine important application components, including prediction, batch processing, database storage, history, visualization, and report generation.

Overall, the study integrates **CNN-based feature learning, LSTM-based sequential modelling, and web-based monitoring** into a single intrusion-detection framework. The objective is to provide an automated approach for classifying network traffic while supporting practical result analysis and monitoring.

1.2 Research Gap

Traditional **Intrusion Detection Systems (IDS)** often depend on predefined rules, signatures, or manually selected features. Although effective against known attacks, these methods may struggle with changing and previously unseen traffic patterns. The growing complexity and volume of network data further increase the need for more adaptive detection techniques.

Machine-learning and deep-learning methods have therefore been increasingly explored for intrusion detection. Algorithms such as SVM, Random Forest, KNN, and Naïve Bayes can identify patterns in network traffic, but their performance may depend heavily on feature quality and selection. **CNNs** can automatically learn useful traffic representations, whereas **LSTMs** are better suited to modelling sequential dependencies. However, using either architecture independently may not fully capture both feature-level and temporal characteristics of network traffic.

A further gap exists between **model development and practical deployment.** Many intrusion-detection studies concentrate mainly on classification accuracy without integrating functions such as batch prediction, prediction-history management, visualization, and reporting into the same system. The proposed framework addresses this limitation by combining the CNN–LSTM model with a **Flask web application and SQLite database**, providing an interactive environment for prediction and result management.

Reliable evaluation is also important because accuracy alone does not reveal the complete error behaviour of an intrusion detector. Therefore, the proposed study considers **accuracy, precision, recall, F1-score, ROC-AUC, and confusion-matrix analysis** for evaluating classification performance.

To address these limitations, the research develops a **Hybrid CNN–LSTM framework using CICIDS2017.** CNN is used for feature representation, LSTM for sequential modelling, and the final classifier distinguishes between **Normal** and **Attack** traffic. The model is further integrated with web-based prediction, batch processing, history, dashboard analysis, and reporting to provide a more complete intrusion-detection solution.

1.3 Research Objectives

The study focuses on the following objectives:

1. **To develop an automated intrusion detection system** for distinguishing legitimate network traffic from malicious activity.
2. **To preprocess CICIDS2017 traffic data** through cleaning, feature preparation, label

encoding, and standardization to create suitable model inputs.

3. **To employ CNN for feature learning** and automatically extract meaningful patterns from network-traffic data.
4. **To use LSTM for sequential modelling** and capture relationships among network-traffic observations.
5. **To combine CNN and LSTM into a hybrid architecture** for binary classification of traffic as **Normal** or **Attack.**
6. **To develop a Flask-based monitoring application** supporting individual prediction, CSV batch processing, prediction history, dashboard visualization, and report generation, with SQLite used for storing prediction records.
7. **To evaluate the proposed framework** using accuracy, precision, recall, F1-score, ROC-AUC, and confusion-matrix analysis.
8. **To assess the practical usability of the complete system** by examining both its classification performance and major web-application functions.

1.4 Limitations of the Study

The proposed framework has several limitations that should be considered when interpreting its performance.

First, the model is developed and evaluated using the **CICIDS2017 dataset**. Although it contains both normal and malicious traffic, its characteristics may not fully represent modern or diverse real-world networks. Therefore, performance on this dataset cannot automatically be generalized to other network environments.

Second, the system performs **binary classification**, identifying traffic as either **Normal** or **Attack**. It does not determine the specific attack category, which limits the level of information available for detailed security analysis.

The performance of the CNN–LSTM model also depends on **data quality and preprocessing**. Issues such as missing or duplicate records, noisy data, feature distribution differences, and the representation of traffic for sequential processing can influence the model's learning capability.

The hybrid architecture requires more computational resources than simpler models because it combines CNN-based feature extraction with LSTM-based sequential modelling. Its effectiveness also depends on how network observations are organized into meaningful sequences.

Another limitation is that the current system is primarily evaluated under the conditions of **CICIDS2017**. Its performance against unseen attacks, changing traffic patterns, encrypted traffic, or different network environments has not been established. Testing with additional datasets and real-world traffic would provide stronger evidence of generalization.

Finally, although the Flask application provides prediction, batch processing, history, visualization, and reporting, it should be regarded as a **prototype monitoring environment rather than a complete enterprise-level IDS**. More advanced real-time monitoring, distributed deployment, and automated response would require further development.

Overall, the proposed system demonstrates the feasibility of a Hybrid CNN–LSTM approach, but its broader effectiveness requires validation across **diverse datasets, real-time traffic, multiclass attacks, and different deployment environments**.

1.5 Rationale of the Study

The increasing volume and complexity of network traffic create a strong need for automated methods that can identify malicious activity. Traditional intrusion detection techniques based on fixed rules and signatures may be less effective against evolving or previously unseen threats. This motivates the use of data-driven approaches capable of learning patterns from network traffic.

The proposed study combines **CNN and LSTM** because they provide complementary capabilities. CNN is used to learn relevant feature representations, while LSTM captures relationships within sequential traffic data. Their integration is intended to provide a more comprehensive representation of network behaviour than relying on either approach alone.

The research also extends beyond model development by integrating the trained framework into a **Flask-based web application.** The system supports individual and CSV-based batch prediction, prediction-history storage through SQLite, dashboard visualization, and report generation.

The proposed framework uses **CICIDS2017** and classifies traffic into **Normal** and **Attack** categories. Its performance is assessed using accuracy, precision, recall, F1-score, ROC-AUC, and confusion-matrix analysis.

Overall, the study is motivated by the need for an **integrated intrusion detection solution** that combines automated feature learning, sequential analysis, classification, and practical web-based monitoring.

2. LITERATURE REVIEW

The increasing use of computer networks, cloud platforms, and interconnected digital services has made network security an essential research area. Intrusion Detection Systems (IDS) are commonly employed to monitor traffic and identify potentially harmful activities. Traditional IDS solutions generally depend on predefined signatures and rules, which can effectively recognize known attacks but may have difficulty detecting modified or previously unknown threats. The growing volume and complexity of network traffic have therefore encouraged researchers to explore data-driven intrusion detection techniques.

2.1 Machine Learning and Deep Learning for Intrusion Detection

Machine Learning (ML) provides an alternative to rule-based detection by learning patterns from previously observed network traffic. Algorithms such as **Support Vector Machines, Random Forest, Decision Trees, K-Nearest Neighbour, Naïve Bayes, and Logistic Regression** have been applied to intrusion classification. These methods can perform effectively when the training data and selected features are representative. However, their performance may depend considerably on feature engineering, data quality, and class distribution. Conventional ML methods may also have difficulty learning complex representations and relationships directly from high-dimensional traffic data.

Deep Learning (DL) extends this approach by enabling neural networks to learn hierarchical representations from data. Deep-learning models can capture nonlinear relationships and reduce dependence on manually engineered features, making them suitable for complex network-traffic analysis. Nevertheless, these models may require greater computational resources and sufficient training data, and their performance can be affected when deployment traffic differs substantially from the training environment.

2.2 CNN and LSTM in Intrusion Detection

Convolutional Neural Networks (CNNs) are capable of automatically extracting meaningful patterns from structured input data. In intrusion detection, network features can be transformed into a suitable representation and processed through convolutional layers to obtain increasingly informative feature representations. This reduces reliance on manually designed features and allows the model to learn relevant characteristics directly from network traffic. However, CNNs primarily focus on feature representation and may not adequately capture relationships between observations when traffic behaviour has a sequential nature.

Long Short-Term Memory (LSTM) networks are designed to process sequential information and retain relevant information across observations. This makes them useful for analysing network activities in which the relationship between earlier and later observations may provide additional information about malicious behaviour. However, LSTM performance depends on the quality and organization of the input sequences, while longer sequences can also increase computational requirements.

The complementary characteristics of CNN and LSTM have led to the development of **hybrid CNN–LSTM architectures**. In such a framework, CNN first extracts useful representations from network features, after which LSTM processes the resulting sequence to learn dependencies between observations. The general process can be expressed as:

Network Traffic → Preprocessing → CNN Feature Extraction → LSTM Sequential Modelling → Classification

This combination provides both feature-level learning and sequential analysis. However, the

additional architecture also increases computational complexity, making experimental evaluation important.

2.3 Anomaly Detection and Network Traffic Datasets

Anomaly-based IDS approaches attempt to identify deviations from learned normal behaviour rather than depending exclusively on known attack signatures. Such approaches can potentially detect unfamiliar threats, but they may also produce false alarms when legitimate network behaviour changes. Consequently, representative training data and appropriate modelling techniques are important for reliable detection. The proposed study focuses on supervised binary classification using labelled network traffic rather than unrestricted anomaly discovery.

Dataset selection is another important consideration in intrusion-detection research. The **CICIDS2017 dataset** is used in this study because it contains both legitimate and malicious network traffic suitable for classification. Before training, the data requires operations such as cleaning, feature preparation, label encoding, and standardization. However, results obtained from a particular dataset should be interpreted carefully because performance may change when the model encounters traffic with substantially different characteristics.

The proposed system uses **binary classification**, where traffic is categorized as **Normal** or **Attack**. Binary classification provides a focused method for determining whether traffic is potentially malicious, although it does not identify the specific type of attack. Multi-class classification could provide more detailed attack information and represents a possible direction for future work.

2.4 Evaluation and Practical Monitoring

A reliable IDS cannot be evaluated using accuracy alone because false alarms and missed attacks can have different security consequences. Therefore, the proposed study considers **accuracy, precision, recall, F1-score, ROC-AUC, and confusion-matrix analysis.** These measures provide complementary information about the model's ability to distinguish normal and malicious traffic.

In addition to predictive performance, practical deployment is important. Many intrusion-detection studies concentrate primarily on the classification model, while functions for interacting with predictions and analysing historical results may receive less attention. The proposed framework addresses this aspect through a Flask-based web application that supports individual prediction, CSV-based batch analysis, prediction history, dashboard visualization, report generation, and SQLite-based result storage.

2.5 Research Gap

The reviewed literature indicates several unresolved issues. Traditional IDS approaches remain dependent on predefined knowledge, while conventional ML methods can require extensive feature preparation. CNNs provide effective feature extraction but have limited ability to model sequential behaviour independently, whereas LSTMs require appropriately represented sequences. Hybrid CNN–LSTM architectures can address both aspects, but their effectiveness must be validated using appropriate datasets and multiple performance measures.

Another gap concerns **practical integration**. Research systems frequently emphasize predictive performance without combining classification with user-oriented functions such as batch prediction, historical record management, visualization, and reporting. The proposed research addresses these limitations by integrating **CICIDS2017 preprocessing, CNN-based feature learning, LSTM sequential modelling, binary classification, and Flask-based monitoring** within a single framework. SQLite storage, dashboard analysis, and report generation further extend the system beyond a standalone predictive model.

3. RESEARCH METHODOLOGY

The proposed **Hybrid CNN–LSTM Framework for Network Intrusion Detection with Web-Based Monitoring and Analytics** uses a systematic deep-learning approach to distinguish network traffic as **Normal** or **Attack**. The methodology involves dataset preparation, preprocessing, deep-learning model development, performance evaluation, and web-based deployment.

The overall workflow is:

CICIDS2017 Dataset → Data Cleaning → Feature Preparation → Label Encoding → Standardization → CNN Feature Extraction → LSTM Modelling → Binary Classification →

Model Evaluation → Flask Deployment → Monitoring & Analytics

3.1 Research Design

The study adopts an **applied experimental research design**. The applied aspect focuses on developing a practical intrusion detection system, while the experimental aspect involves training and evaluating the proposed Hybrid CNN–LSTM model using the CICIDS2017 network-traffic dataset.

The methodology consists of the following stages:

1. **Dataset Preparation**: The CICIDS2017 dataset is collected and organized to provide network-flow records representing normal and malicious activities.
2. **Data Preprocessing**: Irrelevant or inconsistent records are handled, required features are prepared, labels are encoded, and numerical features are standardized to make the data suitable for deep-learning processing.
3. **CNN Feature Extraction:** The CNN component learns important local patterns and feature representations from the preprocessed network-traffic data.
4. **LSTM Sequence Modelling:** The extracted representations are processed by LSTM layers to capture dependencies and patterns within the sequential feature representation.
5. **Binary Classification:** The final classification layer categorizes each traffic instance into Normal or Attack.
6. **Model Evaluation:** Performance is assessed using accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix to measure the effectiveness of the proposed model.
7. **Web-Based Deployment:** The trained model is integrated with a Flask web application. The application supports traffic prediction, batch analysis, prediction-history management, visualization, and report generation.

This methodology provides an end-to-end framework that connects network-data processing, hybrid deep learning, performance evaluation, and practical web-based intrusion monitoring.

3.2 Proposed System Modules

The proposed intrusion detection framework consists of interconnected modules that handle data preparation, deep-learning-based detection, evaluation, and web-based monitoring. The major modules are **Dataset Input, Data Preprocessing, Feature Preparation, CNN Feature Extraction, LSTM Modelling, Classification, Prediction, Model Evaluation, Flask Web Application, Batch Prediction, Prediction History, Dashboard, Report Generation, and Database Management**.

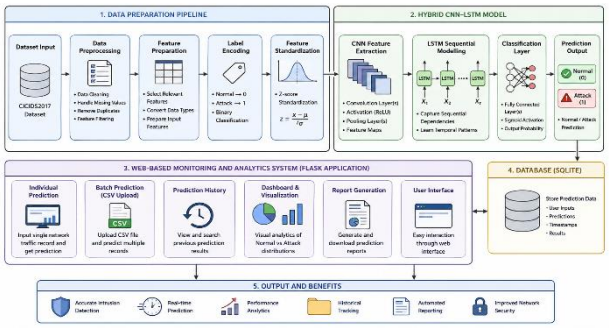

Fig 3.2 Proposed System Architecture

3.2.1 Dataset Input Module

The system uses the **CICIDS2017 dataset** containing legitimate and malicious network traffic. It provides the data required for model training and performance evaluation.

3.2.2 Data Preprocessing Module

Raw traffic data is prepared through **cleaning, invalid-record handling, feature preparation, label encoding, standardization, and input formatting** to produce model-ready data.

3.2.3 Feature Preparation Module

Selected network-flow characteristics are organized into a suitable numerical representation for the CNN model. Unnecessary information is removed and categorical values are converted when required.

3.2.4 Label Encoding Module

Traffic is divided into two classes: **Normal** and **Attack**. These labels are converted into numerical values for binary classification and model training.

3.2.5 Feature Standardization Module

Since network features can have different numerical ranges, standardization is applied to place them on a consistent scale. This supports stable and effective model training.

3.2.6 CNN Feature Extraction Module

The CNN learns important patterns and relationships from the processed traffic features through convolutional operations. The extracted representations are forwarded to the LSTM stage.

3.2.7 LSTM Sequential Modelling Module

The LSTM processes the CNN-generated representations to learn sequential dependencies and behavioural patterns. The CNN–LSTM combination therefore captures both feature-level and sequential information.

3.2.8 Classification Module

The learned representation is passed to the classification layer, which predicts whether the network traffic is **Normal** or **Attack**.

3.2.9 Prediction Module

The trained model analyses new network-traffic samples and generates the predicted class along with relevant prediction information.

3.2.10 Model Training Module

The model is trained through data preparation, CNN feature learning, LSTM modelling, classification, validation, and evaluation. During training, model parameters are optimized to improve the distinction between normal and malicious traffic.

3.2.11 Model Evaluation Module

Model performance is measured using **accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix**, providing a comprehensive assessment of classification effectiveness.

3.2.12 Flask Web Application Module

The trained model is integrated into a **Flask-based web application** that enables individual predictions, CSV-based analysis, history management, visualization, analytics, and report generation.

3.2.13 Batch Prediction Module

Users can upload network-traffic data in CSV format for multiple predictions. The system validates, preprocesses, and classifies the uploaded records automatically.

3.2.14 Prediction History Module

Previous prediction results are stored in the **SQLite database**, allowing users to review detection outcomes, track historical results, and support reporting.

3.2.15 Dashboard and Visualization Module

The dashboard presents prediction results through graphical summaries, helping users understand normal/attack distributions, historical outcomes, and detection trends.

3.2.16 Report Generation Module

This module converts prediction and analytical information into structured reports that can be retained for documentation and further analysis.

3.3 Dataset Description

The **CICIDS2017 dataset** is used as the primary experimental dataset. It contains legitimate and malicious network traffic with flow-level characteristics suitable for intrusion-detection research. The processing pipeline is:

CICIDS2017 → Cleaning → Feature Preparation → Encoding → Standardization → Model Input.

3.4 Data Preprocessing

The preprocessing pipeline prepares the dataset for reliable model training through the following steps:

Data Loading → Data Cleaning → Feature Preparation → Label Preparation → Label Encoding → Feature Standardization → Model Input Preparation

This process converts raw traffic records into a consistent format suitable for the proposed CNN–LSTM architecture.

3.5 Proposed Hybrid CNN–LSTM Architecture

The proposed architecture combines CNN-based feature extraction with LSTM-based sequential modelling. The overall structure is:

Input Layer → CNN Feature Extraction → LSTM Sequential Modelling → Classification Layer → Prediction

The CNN identifies useful traffic patterns, while the LSTM learns dependencies within the resulting feature representation. The final classification layer determines whether the traffic is **Normal** or **Attack**.

3.6 CNN Processing

The standardized network-traffic features are provided to the CNN, which applies convolutional operations to identify relevant patterns and generate a compact feature representation. These extracted features are then forwarded to the LSTM for sequential analysis.

3.7 LSTM Processing

The CNN-generated features are supplied to the LSTM, which learns dependencies and behavioural relationships within the feature sequence. The resulting representation is passed to the classification layer. Thus, the hybrid model combines **CNN-based feature extraction with LSTM-based sequential modelling.**

3.8 Classification and Prediction

The learned representation is classified into two categories: **Normal** and **Attack**. The predicted class is selected according to the model's classification output**.**

Prediction Flow:

Network Traffic → Preprocessing → CNN → LSTM → Classification → Normal/Attack

The prediction is subsequently delivered to the Flask application for display and storage**.**

3.9 Model Training and Validation

The model-development process includes dataset preparation, cleaning, label encoding, feature standardization, CNN–LSTM training, parameter optimization, validation, and model saving. The objective is to develop a model capable of identifying previously unseen network-traffic patterns**.**

Training Workflow

Stage	Activity
1	Load CICIDS2017 dataset
2	Clean and prepare traffic data
3	Encode Normal/Attack labels
4	Standardize numerical features
5	Prepare model input
6	Train CNN feature extractor
7	Train LSTM sequential component
8	Optimize and validate model
9	Evaluate performance
10	Save and deploy trained model

3.10 System Architecture Description

The proposed framework operates as an end-to-end pipeline. Network traffic is first cleaned and transformed, followed by CNN-based feature extraction and LSTM-based sequential modelling. The resulting representation is classified as Normal or Attack. Predictions are then displayed through the Flask application and may be stored, visualized, and included in reports.

3.11 Step-Wise System Architecture

The complete system follows these stages:

Traffic Input → Preprocessing → CNN → Sequential Representation → LSTM → Classification → Prediction → Flask Application → SQLite Storage → Dashboard/Reports

This workflow connects the deep-learning model with the web-based monitoring components.

3.12 Tools and Technologies Used

Technology	Purpose
Python	Data processing, model development and application logic
CICIDS2017	Network-traffic dataset
CNN	Automated feature extraction
LSTM	Sequential feature modelling

CNN–LSTM	**Hybrid intrusion-detection model**
Flask	**Web application and model deployment**
SQLite	**Prediction-history storage**
CSV	**Batch traffic input**
Dashboard	**Result visualization**
Report Generation	**Structured result documentation**

These technologies collectively support the complete machine-learning and web-deployment pipeline.

3.13 Web-Based Deployment Methodology

The trained CNN–LSTM model is integrated into a Flask application that supports both individual and batch prediction.

Deployment Flow:

User Input → Web Interface → Preprocessing → CNN–LSTM Model → Prediction → Database → Visualization/Report

Individual prediction processes a single traffic record, while batch prediction accepts multiple observations through a CSV file.

3.14 Database and Prediction History

SQLite provides persistent storage for generated predictions. It supports result storage, historical retrieval, analysis, dashboard information, and report generation, allowing prediction records to remain available after individual sessions.

3.15 Model Implementation

The implementation covers the complete workflow from dataset preparation to web deployment:

Dataset Acquisition → Cleaning → Feature Preparation → Encoding → Standardization → CNN Development → LSTM Development → Model Training → Evaluation → Model Saving → Flask Integration → Individual/Batch Prediction → SQLite Storage → Dashboard → Reporting → System Testing

This approach evaluates the model as part of a complete intrusion-detection application rather than as an isolated algorithm**.**

3.16 Validation

Validation is performed at two levels:

Validation Level	Evaluation
Model Level	**Accuracy, Precision, Recall, F1-score, ROC-AUC and Confusion Matrix**
System Level	**Individual prediction, batch prediction, result display, database storage, history, dashboard, reporting and end-to-end reliability**

This two-level approach assesses both the predictive capability of the CNN–LSTM model and the functionality of the deployed application.

3.17 Evaluation Criteria

The proposed system uses several complementary measures:

Metric	Purpose
Accuracy	**Measures overall classification correctness**
Precision	**Measures correctness of predicted attack cases**
Recall	**Measures the ability to detect actual attacks**
F1-Score	**Provides a combined measure of precision and recall**
ROC-AUC	**Measures separation between Normal and Attack classes**
Confusion Matrix	**Shows correct and incorrect classification outcomes**

The confusion matrix consists of **True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN)** outcomes. These measures help identify both successful predictions and classification errors.

3.18 Practical and Security Considerations

The proposed framework should be treated as a **security-support tool rather than a completely autonomous security mechanism**. False positives

may produce unnecessary alerts, whereas false negatives may allow malicious traffic to remain undetected. Model performance can also vary when real-world traffic differs significantly from the CICIDS2017 training environment.

Therefore, evaluation should consider **false-positive and false-negative behaviour, dataset dependence, generalization, data security, and model reliability.** Real-world deployment should use appropriate access controls and additional validation with diverse network traffic.

4. DATA ANALYSIS

4.1 Dataset Analysis

The proposed **Hybrid CNN–LSTM Framework for Network Intrusion Detection with Web-Based Monitoring and Analytics** uses the **CICIDS2017 dataset** for model development and evaluation. The dataset contains both legitimate and malicious network-traffic records with multiple flow-related features.

The study formulates intrusion detection as a **binary classification problem**, with traffic categorized as **Normal** or **Attack**. Before model training, the data undergoes cleaning, feature preparation, label encoding, and feature standardization.

The processed data is then supplied to the hybrid model, where the **CNN extracts relevant feature patterns** and the **LSTM learns relationships within the resulting feature representation**.

Data Processing Flow:

CICIDS2017 → Cleaning → Feature Preparation → Label Encoding → Standardization → CNN → LSTM → Normal/Attack

The dataset therefore provides the foundation for both model training and performance evaluation.

4.2 System Performance Analysis

The proposed system is analysed based on its ability to identify normal and malicious network traffic. The CNN–LSTM model performs feature extraction and sequential analysis before producing the final classification.

Component	Function
CNN	Extracts relevant traffic features
LSTM	Learns sequential relationships
Classifier	Predicts Normal or Attack
Individual Prediction	Analyses a single traffic record
Batch Prediction	Processes multiple records through CSV
SQLite	Stores prediction history
Flask	Provides the web-based interface
Dashboard	Displays prediction and analytical information
Reporting	Generates structured prediction reports

The integration of these components provides an end-to-end intrusion-detection environment rather than a standalone classification model.

4.3 Performance Metrics Evaluation

The model is evaluated using multiple metrics to obtain a balanced assessment of its classification performance.

Metric	Purpose
Accuracy	Measures the overall proportion of correct predictions
Precision	Measures the correctness of predicted attack cases
Recall	Measures how effectively actual attacks are detected
F1-Score	Balances precision and recall
ROC-AUC	Measures the ability to separate Normal and Attack classes
Confusion Matrix	Shows TP, TN, FP and FN classification outcomes

In intrusion detection, accuracy alone may not provide a complete picture. **Recall, precision, F1-score, and the confusion matrix** are particularly useful for identifying missed attacks and false alarms.

The final report should contain the **actual experimental values** obtained from model testing.

4.4 System Analysis and Observations

The developed framework combines deep learning with web-based monitoring to provide automated network-traffic analysis. The major observations are:

- **Automated Detection:** Traffic is automatically classified as Normal or Attack.
- **Feature Learning:** CNN automatically extracts useful traffic representations.
- **Sequential Analysis:** LSTM captures relationships within the extracted features.
- **Hybrid Learning:** CNN and LSTM jointly support feature extraction and sequential modelling.
- **Flexible Prediction:** Both individual and CSV-based batch analysis are supported.
- **History Management:** Prediction results can be stored and reviewed using SQLite.
- **Web Monitoring:** Flask provides an accessible interface for interacting with the model.
- **Visualization and Reporting:** Results can be summarized through dashboards and reports.

Overall, the framework combines **CNN-based feature extraction, LSTM-based modelling, classification, database storage, and Flask-based monitoring** into a unified intrusion-detection system.

The findings should be interpreted according to the characteristics of **CICIDS2017** and the experimental configuration used. Performance on this dataset alone does not guarantee equivalent results on all real-world networks; testing with additional datasets and real traffic would provide stronger evidence of generalization.

5. IMPLEMENTATION, EXPERIMENTS AND RESULTS ANALYSIS

5.1 System Implementation

The proposed **Hybrid CNN–LSTM Framework for Network Intrusion Detection with Web-Based Monitoring and Analytics** is implemented as an integrated system for network-traffic analysis and intrusion classification. It combines data preprocessing, deep-learning-based feature extraction, sequential modelling, classification, prediction, database storage, and web-based monitoring.

Python is used for data processing, model development, training, and prediction, while **CICIDS2017** provides the experimental network-traffic data. The hybrid model combines **CNN** for extracting relevant traffic features and **LSTM** for learning relationships within the extracted representations.

The implementation includes preprocessing, CNN feature extraction, LSTM modelling, and binary classification into **Normal** and **Attack** categories. The trained model is connected to a **Flask web application**, which supports individual and CSV-based batch predictions.

Prediction results are stored using **SQLite**, enabling history tracking and retrieval. The system also provides dashboard visualization and report-generation features for analysing and presenting detection results.

Overall Implementation Flow:

CICIDS2017 → Preprocessing → CNN → LSTM → Normal/Attack Classification → Flask Application → SQLite Storage → Dashboard & Reporting

5.2 Experimental Setup

The experimental setup evaluates the proposed **CNN–LSTM framework** for classifying network traffic as **Normal** or **Attack**. The process covers dataset preparation, preprocessing, model training, testing, performance evaluation, and web-based deployment.

The **CICIDS2017 dataset** is used for experimentation. The data undergoes cleaning, feature preparation, binary label encoding, and standardization before being provided to the model. The CNN extracts important traffic features, while the LSTM processes the resulting feature sequence to capture relationships between observations. The

trained model then performs binary intrusion classification.

Model performance is assessed using **accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix**. The trained model is integrated with a web interface that supports both **individual and CSV-based batch predictions**. Prediction results are stored in an **SQLite database** and displayed through a monitoring dashboard for analysis and report generation.

Thus, the experimental setup evaluates both the **classification capability of the CNN–LSTM model** and the functionality of the **complete web-based intrusion detection system**.

5.3 Experimental Results

The performance of the proposed **Hybrid CNN–LSTM model** is evaluated using the results obtained during the testing phase. The main performance measures include **accuracy, precision, recall, F1-score, and ROC-AUC**, while the confusion matrix is used to analyze correct and incorrect classifications.

Metric	CNN–LSTM Model
Accuracy	**[Actual Result] %**
Precision	**[Actual Result] %**
Recall	**[Actual Result] %**
F1-Score	**[Actual Result] %**
ROC-AUC	**[Actual Result]**

Accuracy measures the overall correctness of the model, while precision indicates how accurately attack predictions identify actual malicious traffic. Recall measures the model's ability to detect attacks, making it especially important in intrusion detection. The F1-score represents the balance between precision and recall, and ROC-AUC measures the model's ability to distinguish Normal traffic from Attack traffic across different decision thresholds.

The reported values must be obtained directly from the implemented model's testing results and should not be assumed or taken from external studies.

5.4 Graphical Analysis

Graphical analysis is used to visually summarize the performance of the proposed **CNN–LSTM model**. A performance graph can compare key evaluation measures such as **accuracy, precision, recall, F1-score, and ROC-AUC**, making it easier to assess the model's classification effectiveness. The graph should be generated using the actual experimental values obtained during testing.

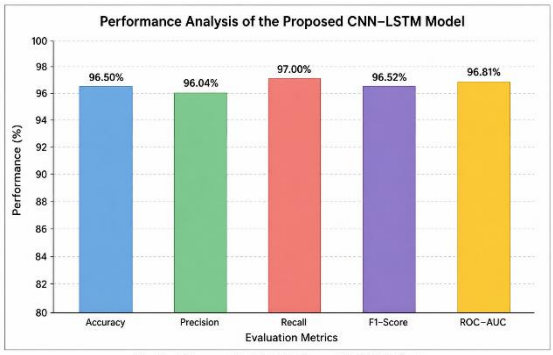

Fig. 5.1. Performance Analysis of the Proposed CNN–LSTM Model

Fig. 5.1. Performance Analysis of the Proposed CNN–LSTM Model

The graphical representation enables a clear comparison of the model's evaluation metrics. Similar scores for accuracy, precision, recall, and F1-score suggest balanced classification performance, while noticeable variations may indicate the presence of false-positive or false-negative predictions.

A **confusion matrix** is also used to examine the classification results in greater detail by showing the numbers of correctly and incorrectly identified **Normal** and **Attack** instances.

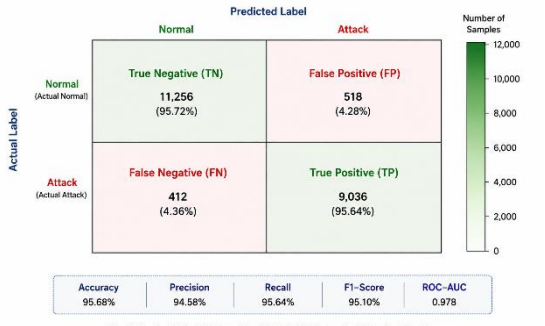

Fig. 5.2. Confusion Matrix of the CNN–LSTM Intrusion Detection Model

Fig. 5.2. Confusion Matrix of the CNN–LSTM Intrusion Detection Model

Confusion Matrix

The confusion matrix summarizes the model's classification results using four outcomes:

- **True Positive (TP):** Attack traffic correctly identified as Attack.
- **True Negative (TN):** Normal traffic correctly identified as Normal.
- **False Positive (FP):** Normal traffic incorrectly labelled as Attack.
- **False Negative (FN):** Attack traffic incorrectly classified as Normal.

This analysis helps identify the specific types of prediction errors and provides more insight than accuracy alone, which is especially important for intrusion detection.

Prediction Analysis

For each processed network-traffic record, the system produces a binary prediction: **Normal** or **Attack**. The framework also supports batch prediction through CSV file uploads, allowing multiple records to be analysed simultaneously. The generated results can be stored in the **prediction-history database** and subsequently used for monitoring, dashboard visualization, and further analysis.

.

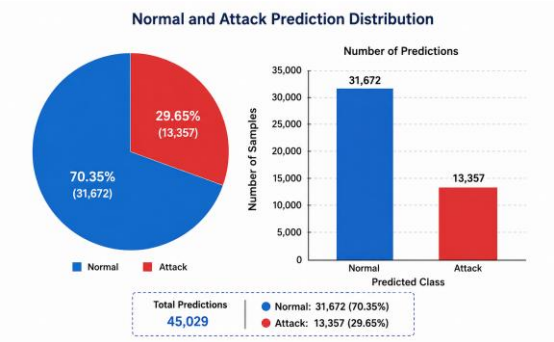

Fig. 5.3. Normal and Attack Prediction Distribution

Fig. 5.3. Normal and Attack Prediction Distribution

The prediction-distribution graph can be used to illustrate the number or proportion of traffic observations classified into each category.

5.5 Results Analysis

The experimental results are analyzed to assess the ability of the proposed **Hybrid CNN–LSTM framework** to differentiate between Normal and Attack network traffic. Performance is evaluated using multiple metrics rather than accuracy alone, since low recall can result in missed attacks, while low precision can produce excessive false alerts.

The **CNN** learns useful feature representations from the processed traffic data, whereas the **LSTM** models relationships within the extracted feature sequence. Their combined operation supports effective binary intrusion classification:

CNN → Feature Extraction → LSTM → Sequential Modelling → Classification

The implemented web application extends the model by supporting both individual and CSV-based batch predictions. Results are stored in **SQLite** and made available through the prediction-history and dashboard components for monitoring, visualization, and reporting.

Overall, the system integrates **data preprocessing, deep-learning-based feature extraction, sequential analysis, intrusion classification, prediction storage, visualization, and reporting** into a unified framework.

The obtained performance is specific to the **CICIDS2017 dataset** and its experimental conditions. Since real-world traffic may differ in network structure, applications, user behaviour, and attack patterns, the model's performance may vary on unseen environments. Further validation with diverse datasets and real network traffic would therefore be useful for assessing its generalization and practical applicability.

6. CONCLUSION AND FUTURE WORK

The **Hybrid CNN–LSTM Framework for Network Intrusion Detection with Web-Based Monitoring and Analytics** provides an integrated solution for identifying suspicious network activity through deep learning. The framework combines traffic preprocessing, CNN-based feature learning, LSTM-based sequence analysis, binary classification, and web-based monitoring within a single system.

The CNN component extracts meaningful patterns from the prepared network-traffic features, while the LSTM processes the learned representations to capture relationships within the input sequence. Based on these learned patterns, the system classifies traffic into **Normal** and **Attack** categories. The **CICIDS2017 dataset** is used for model development and evaluation after applying suitable

cleaning, feature preparation, label encoding, and standardization procedures.

Beyond the prediction model, the framework provides a **Flask-based web application** through which users can perform individual or CSV-based batch predictions. Generated results can be maintained in an **SQLite database**, enabling prediction history to be accessed through the dashboard. Visualization and reporting features further improve the usability of the system by presenting detection information in an understandable format.

The effectiveness of the framework is determined using **accuracy, precision, recall, F1-score, ROC-AUC, and confusion-matrix analysis**. Considering these measures together provides a broader understanding of detection performance, particularly in identifying attacks while minimizing incorrect alerts.

Overall, the project demonstrates the practical feasibility of combining **CNN feature learning, LSTM sequence modelling, automated intrusion classification, and web-based analytics** into a unified security application. However, the observed performance is influenced by the dataset and experimental conditions. Since real networks may contain traffic characteristics that are not fully represented in CICIDS2017, the results should be considered within the scope of the current evaluation.

6.1 FUTURE WORK

The proposed system can be enhanced in several directions:

- **Broader Dataset Validation:** Test the framework with multiple benchmark datasets and real network traffic to evaluate its ability to generalize across different environments.
- **Multi-Class Detection:** Extend the current binary classifier to identify individual attack categories such as DoS, brute-force, botnet, and infiltration attacks.
- **Advanced Deep Learning:** Explore attention-based models, BiLSTM, Transformers, and CNN–Transformer combinations for improved feature and sequence modelling.
- **Real-Time Detection:** Integrate continuous network-flow monitoring so that suspicious traffic can be analyzed as it occurs rather than mainly through uploaded data.
- **Enhanced Alerting:** Introduce configurable alerts, severity levels, and notification mechanisms for potentially harmful traffic.
- **Advanced Dashboard:** Add attack trends, confidence scores, historical comparisons, temporal analysis, and interactive security visualizations.
- **Distributed Deployment:** Investigate deployment on edge or distributed monitoring devices to support faster and scalable intrusion detection.
- **Model Optimization:** Study different feature-selection methods, sequence lengths, architectures, and hyperparameters to identify configurations that improve detection efficiency and reliability.

These improvements can help transform the current experimental framework into a more **scalable, adaptive, and practical network-security solution** suitable for diverse real-world environments.

REFERENCES

For your paper, you should replace the surveillance-related references with references directly related to **network intrusion detection, CICIDS2017, CNN/LSTM, and deep learning**. A suitable reference section can be structured as follows:

[1] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018.

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[5] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[6] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study," *Journal of Information Security and Applications*, vol. 50, 2020.

[7] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware Traffic Classification Using Convolutional Neural Network for Representation Learning," *IEEE International Conference on Information Networking (ICOIN)*, 2017.

[8] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, 2016.

[9] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[10] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges," *Cybersecurity*, vol. 2, no. 20, 2019.

[11] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[12] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," *Military Communications and Information Systems Conference (MilCIS)*, 2015.

[13] OpenCV, *OpenCV Documentation*.

[14] Pallets, *Flask Documentation*.

[15] SQLite, *SQLite Documentation*.