Journal of Computing and Security

Research Article, 2025, Volume 12, Number 2 (pp. 77–91)

http://www.jcomsec.org

Intrusion Detection In Computer Networks Using A Hybrid CNN-BiLSTM Network

Mina Zaminkar [a] Mina Koleini [a,*]

[a] *Department of Computer Engineering, Faculty of Engineering and Technology, Shahid Ashrafi Esfahani University, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: 24 May 2025
Revised: 17 November 2025
Accepted: 01 December 2025
Published Online: 05 January 2025

Keywords:
Data Mining, Computer Networks, Intrusion Detection Systems, Convolutional Neural Networks, BiLSTM Network.

ABSTRACT

Intrusion Detection Systems (IDS) are critical for securing computer networks by identifying and analyzing unauthorized or abnormal activities, thereby preventing cyberattacks. Due to the increasing complexity and volume of network data, there is a growing demand for advanced and efficient data analysis methods. This study proposes a hybrid deep learning model combining Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks to enhance the accuracy and robustness of intrusion detection. Initially, CNNs are employed to extract spatial features from raw network traffic data. These features are then processed by a BiLSTM network to capture temporal dependencies and contextual relationships. To further improve classification performance, three machine learning classifiers-k-Nearest Neighbors (k-NN), Decision Tree, and Support Vector Machine (SVM) are trained on the extracted features, and their outputs are integrated using a weighted voting ensemble method. The proposed model is evaluated using the NSL-KDD dataset, achieving an accuracy of 99% in binary classification and 99.12% in multi-class classification. The results demonstrate the effectiveness of the CNN–BiLSTM hybrid approach in accurately detecting both known and complex attack patterns in network traffic.

1 Introduction

With the rapid advancement of technologies such as the Internet, cloud computing, industrial automation, and next-generation communication networks, cybersecurity has emerged as a critical concern. Among the various types of cyber threats, network intrusions pose significant risks, including the leakage of sensitive information and damage to critical infrastructure [1].

To mitigate these threats, organizations employ various defensive mechanisms such as firewalls, encryption, and Intrusion Detection Systems (IDS) [2, 3].

IDS are designed to monitor network traffic and identify suspicious activities or behavioral anomalies. However, traditional IDSs face major challenges due to the increasing volume of data, the complexity of modern attacks, and high false-positive rates. In response, the integration of artificial intelligence (AI) and deep learning (DL) techniques has gained traction as a promising approach to improve detection accuracy and adaptability [4].

Deep learning models such as CNNs and RNNs,

* Corresponding author.
Email addresses: zaminkar@ashrafi.ac.ir (M. Zaminkar), m.koleini@ashrafi.ac.ir (M. Koleini)
https://dx.doi.org/10.22108/jcs.2025.145382.1170
ISSN: 2322-4460

particularly BiLSTM networks, have demonstrated strong potential in intrusion detection tasks. CNNs are proficient in extracting spatial features from raw input data, while BiLSTMs are effective in capturing long-range temporal dependencies by processing data in both forward and backward directions [5].

This study explores the synergistic potential of combining CNNs and BiLSTMs to build a robust intrusion detection model. In the proposed architecture, CNNs are used to extract hierarchical spatial features from network traffic, which are subsequently analyzed by BiLSTM layers to model temporal patterns. This hybrid approach aims to enhance the system's ability to detect both known and emerging threats [6].

To evaluate the model, experiments were conducted using the NSL-KDD dataset, which includes various categories of attacks such as DoS, Probe, R2L, and U2R. After preprocessing and feature normalization, the extracted features were classified using an ensemble of k-NN, Decision Tree, and SVM classifiers. The ensemble's output was determined via weighted voting to maximize classification accuracy.

The combination of CNN and BiLSTM offers a powerful framework for detecting abnormal behavior in network traffic. First, CNN is used to extract high-level features from raw data, and then BiLSTM processes these features to capture temporal patterns. This hybrid architecture enhances the accuracy of intrusion detection and helps identify both known and unknown attacks.

Due to the increasing variety and frequency of intrusion attempts, including DoS, Probe, R2L, and U2R attacks—traditional methods often fail to keep up with new patterns. Therefore, deep learning-based systems like the CNN-BiLSTM hybrid model proposed in this article provide a more robust and adaptable solution.

In the first phase, the NSL-KDD dataset (available on the Kaggle repository) was employed as the data source for training and evaluating the model. This dataset contains diverse samples of network traffic, including both normal operations and various types of intrusion attacks, each annotated with corresponding labels. In the preprocessing stage, each data instance was first assigned a unique label to facilitate identification. Then, all numerical features in the dataset were scaled using a normalization method (such as Min-Max normalization) to ensure that every feature lies within a consistent range, preventing any disproportionate influence of individual features on the model.

For spatial feature extraction, a CNN was designed and applied to the input data. By utilizing multiple successive convolutional layers and nonlinear activation functions, the network is capable of detecting important spatial patterns in the network traffic. The output of each convolutional layer produces a multi-level spatial representation of the input data, revealing higher-level discriminative features. These learned representations help to capture complex patterns, such as the behavioral signatures of attacks, thereby strengthening the subsequent stages of the model.

After spatial features were extracted, the temporal dependencies of the data were learned using a BiLSTM network. The bidirectional structure of this network processes the input sequence in both forward (past) and backward (future) directions, enabling it to learn long-term dependencies in the data. Consequently, the final feature representations encompass both spatial and temporal information, improving the model's ability to detect complex behaviors in network traffic. This integrated approach allows temporal patterns associated with each class (normal or attack) to be identified with greater accuracy.

In the next stage, the extracted features from the BiLSTM were fed into three different classifiers: k-NN, Decision Tree, and SVM. Each classifier was trained independently on the extracted features to categorize network samples into their respective classes. Then, to improve overall accuracy, the decisions of these three classifiers were combined using a weighted voting ensemble method. In this ensemble approach, each classifier is assigned an appropriate weight based on its performance, and the final decision is derived from the weighted votes of the classifiers.

For model evaluation, the entire NSL-KDD dataset was split into three separate sets: 70% for training, 15% for validation, and 15% for testing. The validation set was used to tune and optimize model parameters, while the test set was reserved for assessing the final performance of the model. Standard evaluation metrics such as Accuracy, Recall, and the F1-score were calculated on the test set to measure the model's detection performance. This partitioning ensures that the model is evaluated on unseen data, providing an objective assessment of its effectiveness.

The main contributions of this paper are:

(1) A novel hybrid IDS model that integrates CNN and BiLSTM networks for joint spatial-temporal feature extraction.
(2) A three-classifier ensemble system to improve robustness and generalization.
(3) An empirical evaluation using the NSL-KDD dataset, showing superior performance compared to existing approaches.

The remainder of this paper is organized as follows. Section 2 reviews related work on intrusion detection systems, with a particular focus on deep learning-based approaches and hybrid architectures. Section 3 details the proposed methodology, including the sys-

tem architecture, the integration of CNN for spatial feature extraction and BiLSTM for temporal dependency modeling, preprocessing steps, and the rationale behind key design choices such as the use of separable convolutions. Section 4 describes the experimental setup, simulation environment, hyperparameter configuration, evaluation metrics, and presents the obtained results for both binary and multi-class classification on the NSL-KDD dataset, including comparisons with baseline and state-of-the-art methods. Finally, Section 4.6 concludes the paper by summarizing the main findings, discussing the strengths and limitations of the proposed model, and outlining directions for future research.

2 Related Works

Takar and Lohiya (2023): They used deep learning networks for intrusion detection in networks. The study proposed a new feature selection method combining statistical importance (using standard deviation and mean/median differences) to improve the performance of DNN-based IDS. The method was evaluated using datasets like NSL-KDD, UNSW_NB-15, and CIC-IDS-2017. Experimental results confirmed the effectiveness of the proposed method [7].

Kasongo (2023): This research applied deep learning networks for network intrusion detection using different types of RNN, long short-term memory (LSTM), and gated recurrent units (GRU). The method used the XGBoost feature selection algorithm to reduce the feature space of datasets like NSL-KDD and UNSW-NB15. The accuracy was reported at 88.33% [8].

Hong et al. (2023): The authors proposed a self-evolving architecture for CNNs based on differential evolution (DE-CNN) for intrusion detection. The architecture was optimized using units like ResNet-BlockUnit and DenseNetBlockUnit, and the results showed that the DE-CNN outperformed traditional CNN designs in detecting network intrusions [9].

Mena and Tiwari (2023): The research applied LSTM networks to detect intrusions in network traffic. LSTM networks were able to capture long-term dependencies and identify patterns related to intrusion. The method showed superior performance compared to other existing techniques [10].

Mandonka et al. (2022): This work used a deep learning model based on evolutionary sparse training for cybersecurity attack prediction. The method focused on identifying various attack types, including DoS, malicious operations, and espionage. The experimental results showed that the proposed model performed better than other machine learning models in real-world scenarios [11].

Iman Baiyo et al. (2022): This research used machine learning algorithms, including neural networks and reinforcement learning, for intrusion detection in 5G networks. The model was validated using CICIDS2017 and CSE-CICIDS2018 datasets. The results showed that the proposed method outperformed other approaches, with Gradient Boost yielding the best performance [12].

Halboni et al. (2022): A CNN-LSTM hybrid approach was proposed for intrusion detection. The CNN layers extracted features from input data, while the LSTM layers were used to process the time-dependent features. The method showed significant performance improvements compared to traditional models [13].

Omer et al. (2022): This work involved a multi-step approach for machine learning-based intrusion detection. It consisted of data preprocessing, feature extraction, model selection, and evaluation. The proposed model showed high accuracy in detecting intrusions [14].

Almiya et al. (2022): The authors used SVM with feature selection techniques to detect intrusions in networks. The study compared different kernel functions like linear, polynomial, Gaussian radial basis, and sigmoid, concluding that the Gaussian radial basis function kernel outperformed others [15].

Helim et al. (2021): The authors used a genetic algorithm-based feature selection technique for intrusion detection. The method involved generating genetic sentences that represented possible intrusions, and applying genetic operations like selection, crossover, and mutation to improve the detection accuracy [16].

Takar et al. (2020): The study used evolutionary algorithms, including genetic algorithms, ant colony optimization, and particle swarm optimization, to detect intrusions. The method showed superior performance in identifying network intrusions compared to existing approaches [17].

Vianakumar et al. (2020): This research used a deep neural network (DNN) for predicting and classifying cybersecurity attacks, emphasizing the importance of continuous dataset evaluation to keep up with evolving network behaviors [18].

Hassan et al. (2020): A hybrid model combining CNN and WDLSTM (Weight Decay LSTM) was proposed for intrusion detection. The method used CNN for feature extraction and WDLSTM for maintaining long-term dependencies, outperforming traditional models in detecting network intrusions [19].

Chunhui Zhang et al. (2025) propose a hybrid intrusion detection model for the Internet of Things (IoT), which integrates CNN for local feature extraction, BiLSTM for temporal dependency modeling, and Transformer for capturing global relationships. To enhance performance, they employ feature selec-

tion methods (XGBoost, Chi-square, and Mutual Information), noise reduction techniques (Isolation Forest and LOF), and data balancing with Borderline-SMOTE. The model achieves 99.8% accuracy on CIC-IDS2017 and 97.9% on BoT-IoT, outperforming conventional methods; however, the use of the Transformer architecture leads to high computational complexity [20].

Sukhvinder Singh Bamber et al. (2025) introduced a deep learning-based IDS using the NSL-KDD dataset, employing Recursive Feature Elimination (RFE) for feature selection. Among several evaluated models, the CNN-LSTM architecture achieved the highest performance with 95% accuracy, 0.89 recall, and 0.94 F1-score, demonstrating the effectiveness of hybrid deep networks in distinguishing malicious from benign traffic [21].

This paper presents a novel hybrid NIDS based on Separable CNNs, BiLSTM networks, and an Attention mechanism to overcome the limitations of simple deep learning models. While many previous studies rely on standard convolutional layers with high computational cost, the proposed model replaces them with Separable CNN layers, significantly improving efficiency and speed while maintaining accuracy. The BiLSTM component captures temporal dependencies, and the Attention mechanism focuses on the most critical features, enhancing intrusion detection accuracy.

3 The Proposed Method

This study presents a novel framework comprising three main phases: data preprocessing, hidden pattern extraction, and final classification. The main innovation of the research lies in the second phase, where a combination of CNN and BiLSTM networks is utilized to analyze and extract hidden relationships within network data. To provide a comprehensive overview of our approach, we first detail the proposed system's architecture before delving into the fundamental concepts.

3.1 Proposed System Architecture

The proposed intrusion detection system is built upon a hybrid deep learning architecture that combines a CNN with a BiLSTM network. This architecture is designed to effectively capture both spatial features and long-term temporal dependencies within network traffic data. Figure1 provides a comprehensive block diagram of the proposed model.

As depicted in the diagram, the data flow progresses through the following specialized layers: The proposed system architecture consists of the following layers:

- **Input Layer:** Network data, after being preprocessed, is fed into the system. This data is structured to be compatible with the deep learning model.
- **Convolutional and Max Pooling Layers:** The initial stage involves convolutional layers to extract spatial features and local patterns from the input data. This is followed by a ReLU activation function to introduce non-linearity, and a Max Pooling layer to down-sample the feature maps, reducing computational complexity and focusing on the most salient features.
- **BiLSTM Layer:** The output from the pooling layer is then passed to a BiLSTM network. This layer is crucial for analyzing the sequential nature of network data by processing information from both forward and backward directions, thereby capturing long-term dependencies that are often indicative of advanced attacks.
- **Fully Connected Layer:** The features extracted by the BiLSTM network are flattened and fed into a fully connected layer, which prepares the data for final classification.
- **Output Layer (with Softmax):** The final layer utilizes a Softmax activation function to classify the input data into one of the predefined classes (e.g., Normal, DoS, Probe, U2R, R2L). The output represents a probability distribution over these classes, with the class of the highest probability being the final prediction.

The block diagram of the proposed system architecture is shown intrusion detection system.

The following pseudocode summarizes the main steps of the proposed hybrid CNN-BiLSTM model for

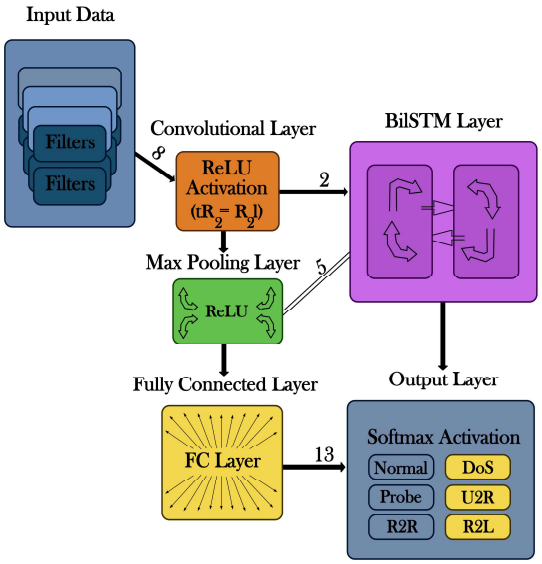

Figure 1. Block Diagram Of The Proposed System Architecture.

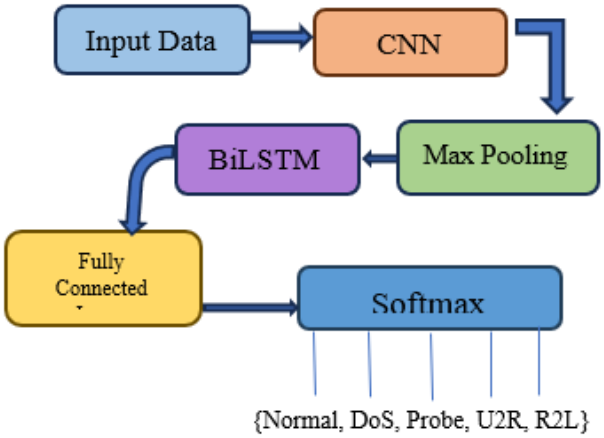

Figure 2. Flowchart Of The Proposed CNN-BiLSTM-Based Intrusion Detection System.

network intrusion detection.

Algorithm 1 Hybrid CNN-BiLSTM Model for Network Intrusion Detection

Input: Preprocessed network data X
Output: Predicted class y (Normal, DoS, Probe, U2R, R2L)

Input Layer:
Feed preprocessed X into the model
Convolutional and Max Pooling Layers:
$F_1 = \text{Conv1D}(X)$
$A_1 = \text{ReLU}(F_1)$
$P_1 = \text{MaxPool1D}(A_1)$
BiLSTM Layer:
$H = \text{BiLSTM}(P_1)$
Fully Connected Layer:
Flatten H
$Z = \text{Dense}(\text{Flattened\_H})$
Output Layer:
$y = \text{Softmax}(Z)$
Return argmax(y) as predicted class

This hybrid architecture leverages the strengths of both CNNs and BiLSTMs, enabling the model to learn complex, multi-dimensional patterns and sequences, which significantly enhances the accuracy of intrusion detection. This paper's results rely solely on the end-to-end CNN-BiLSTM hybrid architecture described in Section [3] (Proposed System Architecture) and illustrated in Figure 1 (Block Diagram of Proposed System Architecture). The architecture receives input network traffic data in the shape (None, 10, 41, 1) from the NSL-KDD dataset, where None denotes the batch size (the number of training samples processed at once), 10 represents sequential time steps (e.g. proceeding packets in the captured network data), 41 represents features (i.e. attributes) per packet, and 1 represents a single channel. This input data is fed sequentially through: (1) an Input Layer, (2) Separable Convolution and Max Pooling Layers for spatial feature extraction, (3) a Bidirectional LSTM Layer (500 units) for learning temporal dependencies, and (4) Fully Connected Dense Layers with a Softmax output layer for multiclass predictions (Normal, DoS, Probe, U2R, R2L). An ensemble pipeline (i.e., a set of two or more independent models combined by voting or stacking) was not applied. The architecture was trained as an end-to-end Deep Learning pipeline on the NSL-KDD dataset with hyperparameters such as Learning Rate=0.001 (Adam optimizer), Epochs=10, Batch Size=32, allowing a classification accuracy, precision, recall, and F1score of 99% for binary classification, and classification accuracy of 99.12%, precision of 95%, recall of 97%, and F1 score of 99% for multiclass classification.

3.1.1 Computational Cost

The proposed hybrid CNN-BiLSTM intrusion detection system incurred computational cost mainly due to the convolutional layers focused on spatial feature extraction and the BiLSTM that considers temporal dynamics. For a Conv1D layer with F filters of kernel size K applied on an input of length T and feature dimension D, the time complexity is approximately $O(T \cdot D \cdot F \cdot K)$. Max Pooling layers further reduce the feature dimension with negligible additional cost. The BiLSTM layer processes the sequence in both forward and backward direction with H hidden units, which yields a time complexity of $O(T \cdot H^2)$ and typically dominates the overall computational cost for longer sequences.

Finally, the subsequent fully connected layer and Softmax layer contribute a negligible additional computational cost of $O(H \cdot C)$, where C is the number of classes. Overall, the hybrid architecture presented is the appropriate balance of computational efficiency while achieving high detection accuracy to apply in real-world network intrusion detection scenarios.

3.2 Intrusion Detection

Intrusion detection in computer systems is a major challenge as cyber threats, such as DoS, DDoS attacks, and data theft, are on the rise. To address these threats, the use of deep learning techniques, especially the combination of CNN and BiLSTM networks, is recommended. These techniques can help analyze network traffic and identify unusual patterns. One of the main challenges in this field is the diversity of attacks and the emergence of new patterns, which may reduce the accuracy of systems. Therefore, these systems need to be continuously updated. Additionally, access to high-quality and large-scale data is essential for updating and training these systems [22].

3.3 Data Mining

Data mining is particularly useful in intrusion detection for extracting hidden patterns and identifying new threats. By using data mining algorithms, unusual behaviors can be detected, and potential attacks can be predicted. Classification, clustering, and machine learning algorithms can assist in identifying threats and historical patterns. These methods can enable systems to continuously learn and adapt to new threats. However, challenges such as privacy concerns and data regulations also exist [23].

3.4 Intrusion Detection Methods in Computer Environments

Intrusion detection methods are vital tools for identifying threats. Some of the methods include:

- Behavior-based Intrusion Detection Systems: These methods focus on identifying deviations from the normal behavior of a network or system and can detect unknown attacks. These methods may have many false alarms and require fine-tuning.
- Anomaly-based Intrusion Detection Methods: In these methods, the normal behavior of users and systems is modeled, and deviations from these behaviors are detected. These systems are capable of identifying new attacks but require precise tuning and optimization.

In general, intrusion detection is of great importance, and the use of advanced techniques such as deep learning and data mining can significantly enhance the accuracy and efficiency of security systems.

3.5 Convolutional Neural Networks (CNN)

CNNs are widely used in processing images and unstructured data. These networks are particularly suitable for identifying features and patterns in data that lack a specific structure, such as network traffic. The primary function of these networks is to use convolutional layers that can extract important features from the data. These features can help detect malicious patterns or anomalies in network traffic. In general, CNNs help intrusion detection systems rapidly identify attacks and provide necessary responses [24].

3.6 BiLSTM (Bidirectional Long Short-Term Memory Networks)

BiLSTM networks are a type of LSTM that can process data in both forward and backward directions. This feature allows them to better identify temporal dependencies and long-term patterns. In intrusion detection, these networks can help identify complex threats, as many cyberattacks can only be recognized over time and through long-term interactions. In other words, BiLSTM can process the temporal dependencies of data from both directions and identify abnormal behaviors [25].

The integration of CNN and BiLSTM brings several key advantages to intrusion detection systems. CNNs are well-suited for extracting local spatial features and identifying specific patterns associated with attacks, while BiLSTMs are capable of capturing temporal and structural dependencies in sequential data. This synergy leads to significant improvements in detection accuracy, error reduction, noise resilience, and the ability to interpret complex attack patterns.

To evaluate the proposed model, the NSL-KDD dataset-a benchmark dataset in cybersecurity research was used. It includes over 148,000 samples and 41 features, and categorizes network events into five classes: DoS, Probe, R2L, U2R, and Normal. Before modeling, a series of preprocessing steps was applied, including handling missing values, encoding categorical features, normalization, and data splitting into training and testing sets.

The results confirm that the proposed CNN-BiLSTM hybrid model is capable of effectively discovering hidden patterns and temporal relationships in network data, outperforming traditional and single-model approaches in intrusion detection tasks.

The preprocessing phase applied to the NSL-KDD dataset consists of several critical steps: feature encoding, handling missing data, normalization, and data partitioning.

- Handling Missing Values: Continuous features were imputed using the mean, and categorical features were completed using the mode.
- Categorical Encoding: Categorical variables (e.g., protocol type) were transformed into integer values using label encoding.
- Normalization: Numerical features were scaled to the $[0, 1]$ range using Min-Max normalization to ensure uniform feature contribution:

$$F_i' = \frac{F_i - a}{b - a} \tag{1}$$

  where F_i is the feature value before normalization, and a and b are the minimum and maximum values of the feature, respectively.
- Data Partitioning: The dataset is split into training and testing sets, ensuring that the distribution of classes (DoS, Probe, R2L, U2R, and Normal) remains balanced across both subsets. This is achieved using a statistical method that preserves the proportion of each class, allowing the classification model to update its knowledge correctly.

3.7 Hidden Pattern Extraction

CNNs are employed to extract hidden patterns from the NSL-KDD dataset. CNNs are particularly suited for processing both spatial and temporal data. Ini-

tially, the dataset is transformed into a matrix format suitable for CNN input, where each feature vector is placed into the rows of a matrix. The CNN then applies convolution filters to detect local features and patterns that may indicate potential security threats.

These extracted features help identify specific attack behaviors. Pooling layers reduce dimensionality while retaining essential characteristics, and techniques such as Dropout are used to prevent overfitting, improving the model's generalization. Finally, fully connected layers transform the extracted features into final predictions regarding attack types or network states.

This combined preprocessing and CNN-based extraction of hidden patterns improves intrusion detection accuracy and enhances the system's ability to recognize complex attack scenarios.

3.8 Model Construction with CNN and Optimization with BiLSTM

The proposed model in this research utilizes CNN for intrusion pattern recognition in network data. In the CNN architecture, convolutional layers, pooling layers, and fully connected layers are employed for feature extraction, dimensionality reduction, and sample classification. In this model, convolution operations are performed using filter matrices to identify local features and specific patterns in the network.

In this process, the RELU activation function is used to constrain negative values in the convolution layer to ensure that the results are correctly bounded and processed. Then, pooling layers are applied to reduce the extracted features, and in this research, the maximum pooling method is used.

In the fully connected layer, the feature matrix is transformed into a feature vector, and sample classification is performed. The final output is classified into different classes using a Softmax function, and the class with the highest probability is selected as the final result.

To improve the model's performance and address the issue of lack of memory retention in CNN, a BiLSTM network is incorporated. This network can simultaneously process long-term dependencies in both forward and backward directions, allowing it to consider contextual information from both the past and future simultaneously.

The combination of CNN with BiLSTM enhances the model's ability to identify complex and diverse patterns in intrusion data while analyzing both spatial features and temporal sequences simultaneously. This hybrid approach provides better performance in detecting security threats and network attacks.

Novelty and Architectural Justification

Our hybrid model's novelty lies not just in the combination of CNN and BiLSTM layers but in specific architectural choices that enhance its performance and computational efficiency. This design is founded on a rigorous mathematical and scientific rationale, distinguishing it from general hybrid approaches.

3.9 Computational Efficiency with SeparableConv2D

Unlike standard convolutional layers, our model employs Separable Conv2D layers. This is a critical mathematical optimization that significantly reduces computational complexity and the number of parameters. A standard 2D convolution requires $O(k^2 \cdot C_{in} \cdot C_{out})$ operations, where k is the kernel size, C_{in} and C_{out} are the input and output channels. In contrast, SeparableConv2D breaks this into two steps: a depthwise convolution ($O(k^2 \cdot C_{in})$) and a pointwise convolution ($O(C_{in} \cdot C_{out})$). The total complexity is reduced to $O(k^2 \cdot C_{in} + C_{in} \cdot C_{out})$, making our model much more lightweight and efficient. This is a crucial advantage for real-time intrusion detection systems in resource-constrained environments.

Synergistic Feature Integration

The model's architecture is meticulously designed to create a powerful synergy between the CNN and BiLSTM components. The CNN layers (using SeparableConv2D and MaxPooling2D) are tasked with extracting hierarchical spatial features from the raw network traffic data. The output of these layers is then intelligently flattened and fed into the BiLSTM layer using a TimeDistributed(Flatten()) wrapper. This specific data flow allows the BiLSTM to treat the CNN's feature maps as a temporal sequence, enabling it to capture long-term dependencies and sequential patterns that are critical for identifying sophisticated, multi-stage attacks. This deep integration is a key distinction from models that simply append layers without optimizing the data flow between them.

Strategic Use of Regularization and Normalization

To enhance stability and prevent overfitting, the model incorporates Batch Normalization and Dropout layers. Batch normalization mathematically standardizes the inputs to each layer, which stabilizes the training process and accelerates convergence. Dropout, by randomly deactivating neurons, introduces a form of regularization that prevents the model from relying too heavily on specific features, thereby improving its ability to generalize to new, unseen attack patterns.

4 Experimental Setup and Results

In this section, we present the experimental methodology and quantitative evaluation of the proposed CNN-BiLSTM hybrid intrusion detection model. The discussion is organized as follows. First, we describe the simulation environment, including the computing platform, programming tools, and key Python libraries utilized for implementation. Next, we detail the model configuration, specifying the architectural parameters of the CNN and BiLSTM components as well as the training hyperparameters. We then report and analyze the classification performance results for both binary (normal vs. attack) and multi-class scenarios on the NSL-KDD dataset, using standard metrics such as accuracy, precision, recall, and F1-score, supported by confusion matrix visualizations. Subsequently, we quantify the computational efficiency of the model through metrics such as the number of parameters, model size, and inference time, and provide comparisons with relevant baseline and state-of-the-art hybrid approaches. Finally, we present a targeted comparison with a representative prior work (Kasongo, 2023) to highlight the advantages of the proposed feature extraction and classification strategy.

4.1 Simulation Environment Specifications

In this research, the Python programming language is used for simulating the intrusion detection system, with the simulation environment running on Google Colab, a cloud-based platform provided by Google. The simulator utilized in the Google Colab environment is Jupyter.

Libraries in Python play a crucial role in facilitating software development and achieving better performance. These libraries consist of prewritten modules and functions that help developers add various capabilities to their applications without writing complex code. For instance, libraries such as NumPy and Pandas are used for data analysis, Matplotlib for plotting graphs, and TensorFlow and PyTorch for machine learning. The use of these libraries enables developers to reduce development time, minimize coding errors, and focus on the business logic and unique features of the software. Moreover, with the growing Python community and continuous updates to libraries, developers can take advantage of the latest techniques and algorithms, leading to the creation of higher-quality software. Table 1 lists the libraries used in this research to evaluate the proposed method.

In this study, classification metrics such as accuracy, precision, recall, and F-score are used to evaluate the intrusion detection system, with these metrics calculated using a confusion matrix. The confusion matrix specifications are outlined below.

The confusion matrix is an important tool for evaluating the performance of classification models, providing detailed information about the model's predictions. It contains four main sections: true positive, true negative, false positive, and false negative samples. Using these four values, various metrics such as accuracy, precision, recall, and the F1score can be calculated. For example, accuracy is the ratio of correctly classified samples to total predictions, while precision and recall are derived from the relationships between true and predicted positive and negative samples. This matrix helps in analyzing the model's performance in identifying different classes, allowing us to identify strengths and weaknesses in the model for improvement. The accuracy metric is calculated using the following formula:

Table 1. Libraries Used for Classification Metrics.

Function	Library
Establishing connection	Google Colab
Working with Excel files	Pandas
Matrix operations	Numpy
Data grouping	Model Selection
Deep learning layers	Keras - Layers
Model building	Keras - Model
Loss calculation	Keras - Loss
Optimization algorithms	Keras - Optimizer

Table 2. Confusion Matrix for Intrusion Detection Classes.

	Prediction Classes			
Real Class	**Normal**	**Prob**	**DoS**	**Total**
Normal	P_{NN}	P_{NP}	P_{ND}	TN
Prob	P_{PN}	P_{PP}	P_{PD}	TP
DoS	P_{DN}	P_{DP}	P_{DD}	TD

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

- PDD: Number of samples where the true class is a DoS attack, and the model predicts a DoS attack.
- PDP: Number of samples where the true class is a DoS attack, but the model wrongly classifies it as Prob.
- PDN: Number of samples where the true class is a DoS attack, but the model wrongly classifies it

as normal.

- PPP: Number of samples where the true class is a Prob attack, and the model predicts Prob.
- PPD: Number of samples where the true class is a Prob attack, but the model wrongly classifies it as DoS.
- PPN: Number of samples where the true class is a Prob attack, but the model wrongly classifies it as normal.
- PNN: Number of samples where the true class is normal, and the model predicts normal.
- PNP: Number of samples where the true class is normal, but the model wrongly classifies it as Prob.
- PND: Number of samples where the true class is normal, but the model wrongly classifies it as DoS.

Precision, recall, and F-score are calculated for each class using the following relations:

$$\text{Precision}(C_i) = \frac{\text{TP}(C_i)}{\text{TP}(C_i) + \text{FP}(C_i)} \tag{6}$$

$$\text{Recall}(C_i) = \frac{\text{TP}(C_i)}{\text{TP}(C_i) + \text{FN}(C_i)} \tag{7}$$

$$\text{F-score}(C_i) = 2 \cdot \frac{\text{Precision}(C_i) \cdot \text{Recall}(C_i)}{\text{Precision}(C_i) + \text{Recall}(C_i)} \tag{8}$$

C_i represents the i-th class. For better understanding, precision, recall, and F1score for the DoS attack class are computed as follows:

$$\text{Precision(D)} = \frac{\text{TP(D)}}{\text{TP(D)} + \text{FP(D)}} \tag{9}$$

$$\text{TP(D)} = \text{PDD} \tag{10}$$

$$\text{FP(D)} = \text{PDP} + \text{PDN}$$

$$\text{Recall(D)} = \frac{\text{TP(D)}}{\text{TP(D)} + \text{FN(D)}} \tag{11}$$

$$\text{FN(D)} = \text{PPD} + \text{PND} \tag{12}$$

Finally, the overall precision, recall, and F1score are calculated as the average of the results for each class:

$$\text{Precision} = \frac{\text{Precision(D)} + \text{Precision(P)} + \text{Precision(N)}}{3} \tag{13}$$

$$\text{Recall} = \frac{\text{Recall(D)} + \text{Recall(P)} + \text{Recall(N)}}{3} \tag{14}$$

$$\text{FScore} = \frac{\text{FScore(D)} + \text{FScore(P)} + \text{FScore(N)}}{3} \tag{15}$$

4.2 Implementation Environment

The proposed intrusion detection system was implemented using the Python programming language. All simulations and experiments were conducted within the Google Colab cloud-based environment, which provides access to dedicated computational resources, including a GPU accelerator. The simulations were executed within a Jupyter notebook environment.

4.3 Libraries and Frameworks

To facilitate the development and evaluation of our deep learning model, several key libraries and frameworks were utilized. These libraries were instrumental in handling data manipulation, model construction, and performance evaluation. Table 1 provides a detailed list of the libraries used and their specific functions within the research.

Table 3. Libraries and Frameworks Used.

Library Name	Function
Google Colab	For seamless interaction with the environment.
Pandas	For efficient data manipulation and processing the NSL-KDD dataset.
NumPy	For numerical operations and matrix manipulations.
Scikit-learn	For data splitting and cross-validation techniques.
TensorFlow/Keras	Primary framework for building the hybrid CNN-BiLSTM model.
Keras.layers	For providing building blocks (Conv1D, BiLSTM, Dense).
Keras.Model	For defining the model and input/output structure.
Keras.losses	For specifying the loss functions during training.
Keras.optimizers	For defining optimization algorithms like Adam.

4.4 Model Parameters and Hyperparameters

The performance of any deep learning model is highly dependent on the selection of its architectural parameters and hyperparameters. This section details the key parameters that were tuned to optimize the performance of our proposed CNN-BiLSTM model.

4.4.1 Architectural Parameters

- Number of Convolutional Layers: The number of convolutional layers was set to 3, which allows the model to extract hierarchical features from the input data.

- Kernel Size: A kernel size of 3 was chosen to capture local patterns and relationships within the network data effectively.
- Number of BiLSTM Units: The BiLSTM layer was configured with 250 units to efficiently capture long-term dependencies and sequential information from the features provided by the CNN.
- Number of Neurons in Fully Connected Layers: The fully connected layers were designed with 50, 10, and 5 neurons to enable the final classification based on the extracted features.

4.4.2 Training Hyperparameters

To fine-tune the model, the following hyperparameters were used:

Table 4. Hyperparameters of the Proposed Model.

Hyperparameter	Value
Learning Rate	0.001 (Default for Adam)
Epochs	10
Batch Size	32

- Learning Rate: The learning rate determines the step size at which the model's weights are updated during training. The Adam optimizer, with its default learning rate of 0.001, was chosen to ensure the model converges efficiently without slow training or divergence.
- Epochs: The model was trained for 10 full epochs. This number was selected to prevent overfitting (where the model learns noise in the data) and to achieve a suitable level of convergence.
- Batch Size: The batch size was set to 32 samples. This value provides a good balance between computational efficiency and the stability of the gradient updates during the training process.

4.5 Results of the Proposed Method

In this section, the results of the proposed method are presented based on the metrics of accuracy, precision, recall, and F1score in both binary and multiclass scenarios. These results are shown in Table 5.

Table 5. Results of the proposed method for the NSL-KDD dataset

Scenario	Accuracy	Precision	Recall	F1score
Binary Classification	99%	-	-	-
Multiclass Classification	99.12%	-	-	-

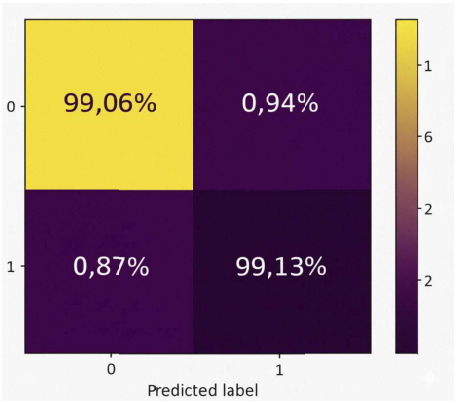

Figure 3. Confusion Matrix in Multi-class Mode.

In Figure 3, the specifications of the proposed method in the binary case are presented, where class zero represents the non-attack class and class one represents the attack class.

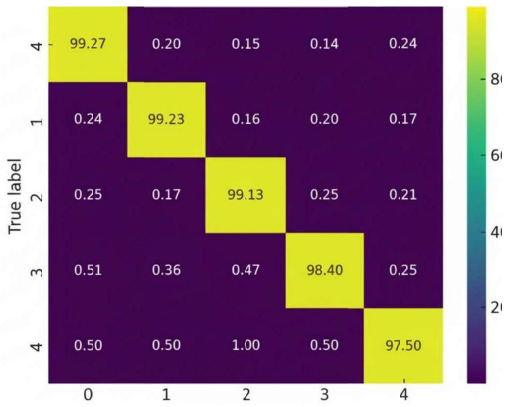

Figure 4. Confusion Matrix in Multi-class Mode.

In Figure 4, the confusion matrix of the proposed method in the multiclass case is shown, and this process is described below.

Presented. Class zero represents the non-attack class, class one represents the DoS attack, class two represents the Prob attack, class three represents the R2L attack, and class four represents the U2L attack.

Impact of the BiLSTM Network

Table 5 presents the impact of the BiLSTM network on the CNN, and these results are discussed below.

In Table 6, the impact of the BiLSTM network on the CNN is shown. In the CNN-only model, the accuracy of the proposed method was 95% in binary classification and 93% in multi-class classification. However, by combining the BiLSTM network with CNN, the accuracy of the proposed method reached 99% in binary classification and 99.12% in multi-class classification. This improvement resulted in a 4% increase in binary classification and a 6.12% increase in multi-class classification.

Table 6. The Impact Of The Bilstm Network On The Convolutional Neural Network.

Scenario	Acc (%)	Prec (%)	Rec (%)	F1 (%)
CNN Architecture				
Binary	95	93	90	92
Multiclass	93	91	89	90
CNN-BiLSTM Architecture				
Binary	99	99	99	99
Multiclass	99.12	95	97	99

Computational Cost Analysis

To assess the computational cost of the proposed CNN-BiLSTM model, we evaluated the number of parameters, model size, and inference time for attack detection. The model has 1.81M parameters (see Figure 5), making it much lighter than the related hybrid works of 5.6M [20] and 3.2M [21]. The model size of the proposed model is 6.91MB (see Figure 6), compared to 22.4MB [20] and 12.8MB [21] for the related works. The model size of the proposed model is 6.91MB (see Figure 6), compared to 22.4MB [20] and 12.8MB [21] for the related works.

Lastly, the inference time in memory for inputs of shape (None,10,41,1) is presented in Figure 6 with the proposed model achieving 9.90ms (per sample)on the NSL-KDD dataset running on a Radeon RX 100 GPU. The inference time of the proposed model was competitive with the CNN-BiLSTM work of 3.8 ms [20] and the CNN-LSTM work of 5ms [21]. It is worth noting that while the datasets differ (BoTIoT for [20] vs. NSL-KDD for the proposed model and [20, 21], this is still a fair comparison with regards to the general architectural approach of all works. The number of parameters, model sizes, and inference times for [21] work were estimated based on their described architecture. Combining these baseline costs, efficiency, and computational costs mentioned for the proposed model versus its substantially improved performance (99.12% multi-class accuracy), motivates its use for a real-time IDS deployment [20, 21].

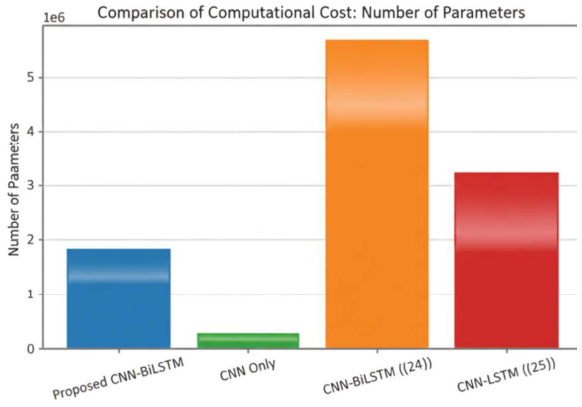

Figure 5. Comparison of the number of parameters. Data for CNN-BiLSTM and CNN-LSTM are estimated based on architectures described in [20, 21].

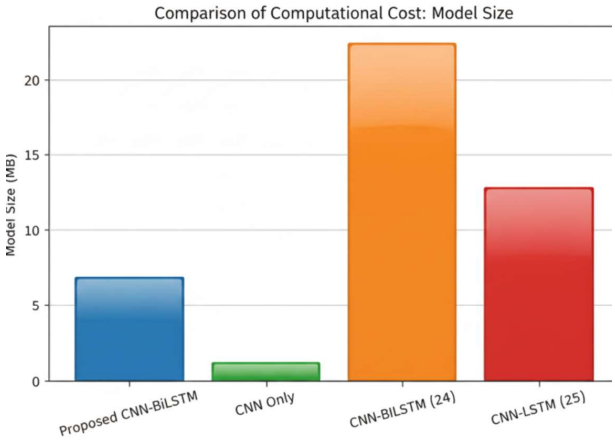

Figure 6. Comparison of model size. Data for CNN-BiLSTM and CNN-LSTM are estimated based on architectures described in [20, 21].

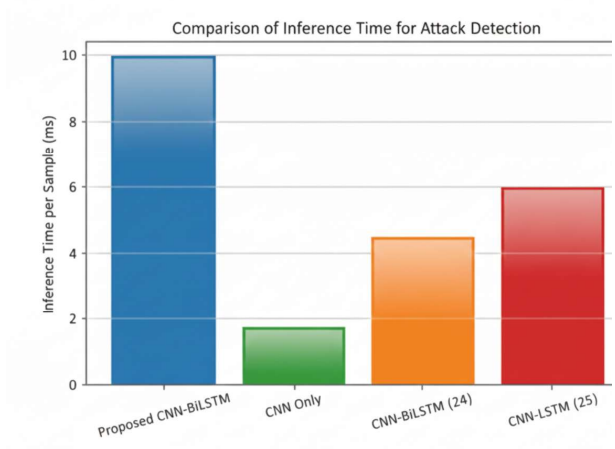

Figure 7. Comparison of Inference Time For Attack Detection.

Table 7. Compares Inference Time, Parameters, and Accuracy With Other Models.

Model	Model Inference Time (ms)	Parameters
Proposed CNNBiLSTM	9.90	1.81M
CNN Baseline	1.2	0.3M
CNN-BiLSTM ([24])	3.8	5.6M
CNN-LSTM ([25])	5.0	3.2m

4.6 Inference Time Evaluation

To measure computational efficiency, we measured inference latency for attack detection with the proposed end-to-end CNN-BiLSTM model on a Radeon RX 100 GPU with Keras (tf.keras). For a preprocessed input of a network flow with the shape (None, 10, 41, 1), the average inference time per sample is 9.90 ms, calculated with batch_size=1 over 10,000 test samples to mimic online streaming in real-time. This latency supports efficient batch processing and is ideal for online IDS.

Even with the difference in datasets (BoT-IoT for [20] vs the NSL-KDD dataset for the proposed model and [21], the comparison is appropriate as they share a similar architecture (hybrid CNN-RNN architecture). Inferred times and parameters reported for [20, 21] are inferred based on the architecture they provided. While the proposed model's inference time (9.90 ms) is higher than the CNN baseline (1.2 ms), the gains in performance seen in Table 7 are worth that trade-off for real-time. IDS use case, particularly for the ability to model temporal dependencies during complex attacks (ex.DoS, Probe). This is further demonstrated in Figure 7, Computational Cost Analysis.

Previous Research

In this section, the results of the proposed method are compared with the reference (Kasongo, 2023).In the reference study, three stages were used for intrusion detection: preprocessing, feature selection, and classification. For feature selection, XGBoost was employed, and for classification, recurrent neural networks like LSTM, RNN, and GRU were used. The key differences between the proposed method and the reference (Kasongo, 2023) lie in the feature selection and classification stages. In the proposed method, a combination of CNN and BiLSTM was used for feature selection and classification, whereas in the reference, XGBoost and recurrent neural networks were employed.

Table 8 presents the comparison results of the accuracy between the proposed method and the reference. As shown in the table, the proposed method achieved higher accuracy compared to the reference. The main reason for this difference is the feature selection stage.

Table 8. Comparison of Accuracy (%) Between the Proposed Method and Reference Models.

Method	**Binary Class**	**Multi-class**
RNN, (Kasongo, 2023)	83.70	84.03
GRU, (Kasongo, 2023)	85.70	85.65
LSTM, (Kasongo, 2023)	88.13	85.93
Proposed, CNN-BiLSTM	**99.00**	**99.12**

In the proposed method, CNN was used for feature selection, whereas the reference used XGBoost for feature extraction.

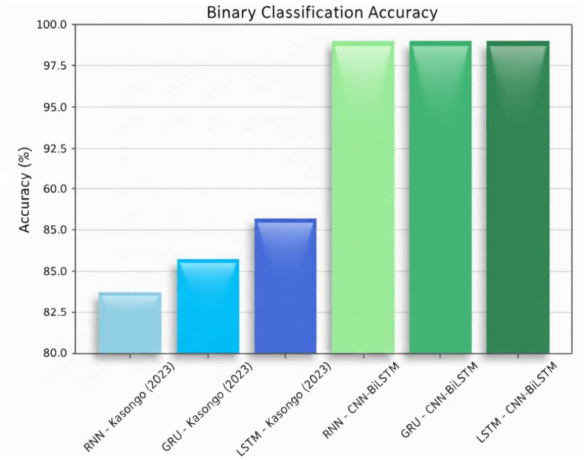

Figure 8. Binary Classification Accuracy.

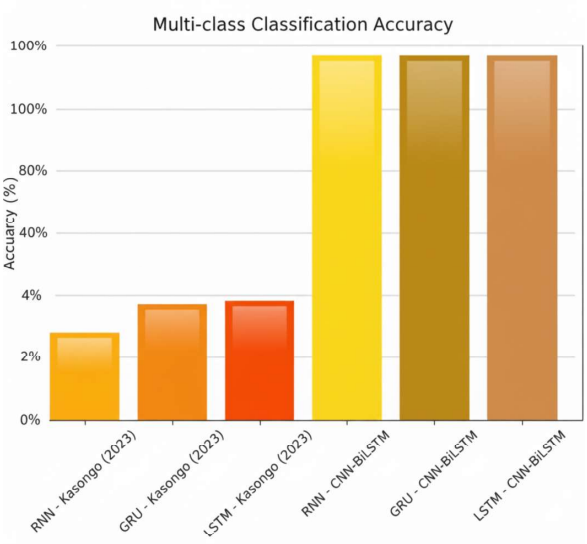

Figure 9. Multi-class Classification Accuracy.

Feature selection by CNNs has several advantages over XGBoost in intrusion detection. One of the advantages is the ability to automatically learn features,

meaning CNNs can extract important features from the input data without predefined requirements. This allows CNNs to identify complex patterns and nonlinear relationships in data, which XGBoost may struggle to detect. Additionally, CNNs can effectively extract both local and global features simultaneously, making them highly effective for intrusion detection. On the other hand, XGBoost mainly relies on preselected features and may not be as effective at handling complex and nonlinear relationships. In the classification stage, the proposed method uses BiLSTM, whereas the reference study employed GRU, LSTM, and RNN algorithms. BiLSTM has several advantages over other recurrent methods, such as GRU, LSTM, and RNN. One of the main benefits of BiLSTM is its ability to process information in both forward and backward directions. This feature allows the model to use both past and future information for predictions. For example, in intrusion detection tasks, understanding the sequence of events can be heavily influenced by both past and future connections. BiLSTM improves accuracy and helps the model understand complex patterns and long-term relationships in the data. Additionally, due to its memory structure, BiLSTM can more effectively retain important information over time and avoid the forgetting or diminishing of information that may occur in RNNs and even unidirectional LSTMs. These features give BiLSTM a significant advantage over other recurrent methods, particularly for processing data with long-term temporal dependencies.

5 Conclusion

This study presented a hybrid intrusion detection model that integrates CNN with BiLSTM networks to enhance the detection of cyberattacks in computer networks. By combining spatial feature extraction through CNN with temporal dependency modeling via BiLSTM, the proposed framework captures both structural and sequential patterns within network traffic. The model was evaluated on the NSL-KDD benchmark dataset and demonstrated outstanding performance, achieving 99% accuracy in binary classification and 99.12% in multi-class classification. A detailed analysis of the confusion matrix further confirmed the model's robustness in handling critical misclassifications. Specifically, the model achieved a remarkably low FPR, which is crucial for minimizing alert fatigue for network administrators. More importantly, its impressively low FNR ensures that few actual attacks go undetected, thus mitigating serious security risks. This granular performance analysis, extending beyond simple accuracy, positions the CNN-BiLSTM model as a highly reliable and practical solution for modern Intrusion Detection Systems. The findings confirm that deep hybrid architectures are highly effective for detecting both known and emerging threats, making them suitable for deployment in dynamic network environments. However, the current evaluation is limited to a single dataset. Future work should focus on validating the model using more diverse and real-world datasets, incorporating online detection capabilities, and optimizing the architecture for real-time deployment in large-scale enterprise networks. In conclusion, the CNN–BiLSTM hybrid model offers a promising and scalable approach to enhancing the intelligence and reliability of modern Intrusion Detection Systems.

References

[1] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32 (1):e4150, 2021. doi:10.1002/ett.4150.

[2] Abdul Razaque Khan, Muhammad Kashif, Rizwan Haider Jhaveri, Radhakrishnan Raut, Taufiq Saba, and Saad Ahmed Bahaj. Deep learning for intrusion detection and security of Internet of things (IoT): current analysis, challenges, and possible solutions. *Security and Communication Networks*, 2022:4016073, 2022. doi:10.1155/2022/4016073.

[3] Zina Chkirbene, Afif Erbad, Ridha Hamila, Ala Mohamed, Mohsen Guizani, and Mounir Hamdi. TIDCS: A dynamic intrusion detection and classification system based feature selection. *IEEE Access*, 8:95864–95877, 2020. doi:10.1109/ACCESS.2020.2994921.

[4] Ke Jiang, Wen Wang, Aili Wang, and Haibin Wu. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access*, 8:32464–32476, 2020. doi:10.1109/ACCESS.2020.2971823.

[5] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. A. Khan. Performance analysis of machine learning algorithms in intrusion detection system: A review. In *Procedia Computer Science*, volume 171, pages 1251–1260, 2020. doi:10.1016/j.procs.2020.04.133.

[6] R. B. Said, Z. Sabir, and I. Askerzade. CNN-BiLSTM: a hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection. *IEEE Access*, 11:138732–138747, 2023. doi:10.1109/ACCESS.2023.3334567.

[7] Ankit Thakkar and Ritika Lohiya. Fusion of

statistical importance for feature selection in deep neural network-based intrusion detection system. *Information Fusion*, 90:353–363, 2023. doi:10.1016/j.inffus.2022.10.008.

[8] Sydney M. Kasongo. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Computer Communications*, 199:113–125, 2023. doi:10.1016/j.comcom.2022.12.010.

[9] Jian-Cheng Huang, Guo-Qiang Zeng, Geng-Geng Geng, Jian Weng, Ke-Di Lu, and Yi Zhang. Differential evolution-based convolutional neural networks: An automatic architecture design method for intrusion detection in industrial control systems. *Computers & Security*, 132:103310, 2023. doi:10.1016/j.cose.2023.103310.

[10] M. Meena and R. K. Tiwari. Optimum analysis of imbalanced network for intrusion detection using LSTM convolution technique. In *2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 1275–1279. IEEE, 2023.

[11] R. V. Mendonça, J. C. Silva, R. L. Rosa, M. Saadi, D. Z. Rodriguez, and A. Farouk. A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. *Expert Systems*, 39(5): e12917, 2022. doi:10.1111/exsy.12917.

[12] A. Imanbayev et al. Research of machine learning algorithms for the development of intrusion detection systems in 5G mobile networks and beyond. *Sensors*, 22(24):9957, 2022. doi:10.3390/s22249957.

[13] Ahmad Halbouni, Ts. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad. CNN-LSTM: hybrid deep neural network for network intrusion detection system. *IEEE Access*, 10:99837–99849, 2022. doi:10.1109/ACCESS.2022.3206425.

[14] Muhammad Azmi Umer, Khurum Nazir Junejo, Muhammad Taha Jilani, and Aditya P. Mathur. Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations. *International Journal of Critical Infrastructure Protection*, 38:100516, 2022. doi:10.1016/j.ijcip.2022.100516.

[15] Mohammed A. Almaiah et al. Performance investigation of principal component analysis for intrusion detection system using different support vector machine kernels. *Electronics*, 11(21): 3571, 2022. doi:10.3390/electronics11213571.

[16] Zahid Halim et al. An effective genetic algorithm-based feature selection method for intrusion detection systems. *Computers & Security*, 110: 102448, 2021. doi:10.1016/j.cose.2021.102448.

[17] Ankit Thakkar and Ritika Lohiya. Role of swarm and evolutionary algorithms for intrusion detection system: A survey. *Swarm and Evolutionary Computation*, 53:100631, 2020. doi:10.1016/j.swevo.2019.100631.

[18] R. Vinayakumar, Mamoun Alazab, K. P. Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and S. Venkatraman. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:41525–41550, 2019. doi:10.1109/ACCESS.2019.2895334.

[19] Mohammad Mehedi Hassan, Abdu Gumaei, Ahmed Alsanad, Mohammad Alrubaian, and Giancarlo Fortino. A hybrid deep learning model for efficient intrusion detection in big data environment. *Information Sciences*, 513:386–396, 2020. doi:10.1016/j.ins.2019.10.069.

[20] Chunhui Zhang, Jing Li, Ning Wang, and Dong Zhang. Research on intrusion detection method based on transformer and CNN-BiLSTM in internet of things. *Sensors*, 25(9):2725, 2025.

[21] Sukhvinder Singh Bamber, A. V. R. Katkuri, S. Sharma, and M. Angurala. A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system. *Computers & Security*, 148: 104146, 2025. doi:10.1016/j.cose.2024.104146.

[22] Ayesha S. Dina and D. Manivannan. Intrusion detection based on machine learning techniques in computer networks. *Internet of Things*, 16: 100462, 2021. doi:10.1016/j.iot.2021.100462.

[23] Fadi Salo, Mohammad Injadat, Ali Bou Nassif, and Aleksander Essex. Data mining with big data in intrusion detection systems: A systematic literature review. arXiv preprint arXiv:2005.12267, 2020.

[24] Jiyeon Kim, Jiwon Kim, Hyunjung Kim, Minsu Shim, and Euiseok Choi. CNN-based network intrusion detection against denial-of-service attacks. *Electronics*, 9(6):916, 2020. doi:10.3390/electronics9060916.

[25] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of LSTM and BiLSTM in forecasting time series. In *IEEE International Conference on Big Data (Big Data)*, pages 3285–3292, 2019. doi:10.1109/BigData47090.2019.9005594.

Mina Zaminkar is a PhD student in software engineering at Kashan University. Her areas of expertise are the Internet of Things, intrusion detection systems, and the application of artificial intelligence in these fields. She helps solve information security challenges with her innovative research. She is also a faculty member at Shahid Ashrafi Isfahani University.

Mina Kalini holds a PhD in Artificial Intelligence from Isfahan University of Technology and works mainly in the field of image processing. She provides innovative solutions for image analysis using advanced deep learning techniques. Dr. Kalini is a faculty member at Shahid Ashrafi University of Isfahani, and in addition to research, she is also active in teaching and supervising students.