Performance Evaluation of Deep Learning Models for Sequence-Based Intrusion Detection

Lahcen Idouglid, Said Tkatek and Khalid Elfayq

Computer Sciences, Research Laboratory, Ibn Tofail University Kenitra, Morocco
lahcen.idouglid@uit.ac.ma, said.tketek@uit.ac.ma, khalid.elfayq@uit.ac.ma

Abstract: The rapid evolution of cyber threats and the increasing complexity of network-connected devices have expanded the need for robust Intrusion Detection Systems (IDS). This paper evaluates the effectiveness of five deep learning models—MLP, LSTM, CNN, RNN, and DNN— for host-based intrusion detection using the ADFA-LD dataset, this study compares models across metrics such as accuracy, precision, recall, and F1-score, highlighting the superiority of sequence-based models such as LSTM and RNN in detecting complex attack patterns. The results demonstrate that while RNN and LSTM provide the highest accuracy, they come with higher computational costs. The findings contribute to the growing field of IDS, offering insights into balancing performance and scalability in real-world applications.

Keywords: Detection Systems (IDS), Deep Learning, ADFA-LD Dataset, Host-Based Intrusion Detection, LSTM, Cybersecurity;

1. Introduction

The growing number of network-connected devices has greatly widened the attack surface for cyber threats, making intrusion detection a crucial component of modern cybersecurity. Intrusion Detection Systems (IDS) are developed to identify unauthorized access and malicious activities within a network or host system, thereby safeguarding data integrity and security.[1], [2]. Conventional IDS methods, such as signature-based approaches, have faced challenges in keeping pace with the increasing sophistication of modern cyber-attacks, particularly due to their inability to generalize beyond predefined attack signatures. This limitation has driven research towards machine learning-based IDS, which are capable of learning from data to detect anomalies and previously unseen threats[3].

Recently, deep learning, a subset of machine learning, has become a powerful tool for intrusion detection, owing to its ability to autonomously learn intricate patterns from large datasets. Deep learning models, including Multi-Layer Perceptrons (MLP), Long Short-Term Memory (LSTM) networks, Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and Deep Neural Networks (DNN), have demonstrated significant potential in identifying intrusions by effectively utilizing the sequential and high-dimensional characteristics of the data [4], [5]. These models are particularly skilled at analyzing system call sequences, which are commonly used to represent both normal and malicious behavior in intrusion detection tasks [6].

The emergence of the Industrial Internet of Things (IIoT) and Industry 4.0 has further complicated the landscape of intrusion detection. IIoT has revolutionized industries such as manufacturing, where interconnected devices and systems work together in smart factories to enhance productivity, enable real-time monitoring, and optimize operations [7]. However, this interconnectivity introduces new vulnerabilities as networks grow in size and complexity, making them susceptible to a wider array of cyber-attacks. Traditional IDS solutions often fall short in these environments, particularly when dealing with high-velocity data streams and real-time decision-making that is essential in IIoT systems [8], [9].

Deep learning models have shown great promise in these environments, providing scalability, flexibility, and the capability to process unstructured data in real-time. In smart factories, these models enable more robust IDS by capturing complex temporal relationships within system data, detecting subtle patterns indicative of intrusion, and dynamically responding to evolving threats. Architectures like LSTM and RNN excel at processing the

Received: August 27th, 2024. Accepted: March 29th, 2025
DOI: 10.15676/ijeei.2025.17.1.5

sequential nature of the data generated in IIoT environments, making them highly effective for anomaly detection and intrusion detection tasks [10].

The ADFA-LD dataset has become a benchmark in the field of host-based intrusion detection. It consists of system call traces representing both normal activities and various types of attacks, such as Hydra and Meterpreter. Due to its sequential nature and diverse attack types, ADFA-LD presents unique challenges for IDS models, making it an ideal dataset for testing the efficacy of deep learning methods [11]. The sequential characteristics of the dataset, coupled with the need for detecting subtle anomalies, make architectures like LSTM and RNN particularly well-suited for this task, as they excel in modeling temporal dependencies [12].

In this paper, we investigate the performance of five distinct deep learning architectures—MLP, LSTM, CNN, RNN, and DNN—on the ADFA-LD dataset. Each model is evaluated based on metrics such as accuracy, precision, recall, and F1-score, with the goal of identifying the most effective approach for host-based intrusion detection. Our findings add to the expanding research efforts focused on enhancing cybersecurity defenses through advanced machine learning techniques [5], [13].

2. Related Work

Intrusion Detection Systems (IDS) have been a cornerstone of cybersecurity for decades, designed to detect malicious activities within networks or hosts. Traditional IDS techniques are generally divided into two categories: signature-based and anomaly-based approaches. Signature-based IDS operate by detecting known threats through the comparison of observed activities against a database of predefined attack signatures. While this method is highly effective against known threats, it struggles to identify zero-day attacks or novel malicious activities that do not match existing signatures [6], [10].

The other hand, anomaly-based IDS seek to identify deviations from established behavioral norms. These systems construct models of "normal" behavior based on historical data and flag any significant deviations as potential intrusions. This approach allows for the detection of previously unknown attacks, but it often leads to higher false positive rates, as accurately defining normal behavior across various environments can be challenging [14]. Anomaly-based techniques have traditionally relied on statistical models, clustering algorithms, or rule-based systems to detect outliers or irregular patterns [1], [15].

With the increasing complexity and scale of networked environments, traditional IDS methods have struggled with scalability, adaptability, and accuracy. This has prompted a shift toward machine learning techniques for IDS. Supervised learning models such as decision trees, support vector machines (SVMs), and random forests have been employed to automate the detection process by learning patterns from labeled data. These models offer improved handling of larger datasets and more complex patterns compared to traditional methods, enhancing both detection rates and adaptability to emerging attack vectors [3], [10].

In recent years, deep learning has emerged as a potent tool for advancing IDS, offering the capability to automatically learn complex features from raw data without requiring manual feature engineering. Architectures such as Long Short-Term Memory (LSTM) networks, Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN) have been extensively used in IDS research due to their superior ability to capture non-linear relationships, model sequence dependencies, and handle high-dimensional data [11], [16], [17]. CNNs have been particularly effective in extracting spatial features from network traffic data, such as packet headers or flow statistics, while LSTMs and RNNs excel at processing sequential data, such as system call traces or time-series network traffic. For instance, RNNs and LSTMs have demonstrated significant advantages in modeling temporal dependencies in system call sequences, which are key indicators of intrusions[5]. These models have shown success in detecting complex attacks, such as advanced persistent threats (APTs) and multi-step intrusions, that traditional and shallow machine learning models struggle to identify [7].

However, deep learning models present certain challenges. One of their primary limitations is the need for large labeled datasets for effective training. Moreover, these models are often

perceived as black boxes, making it difficult to interpret the decision-making process, which can hinder transparency and trust in their outcomes, which can hinder trust in their deployment for security-critical applications. Furthermore, the computational complexity and resource demands of training deep learning models are significantly higher compared to traditional methods, raising concerns about their scalability in real-time detection scenarios[18], [19].

Despite these challenges, recent advancements in hardware acceleration and model optimization techniques have alleviated many of these issues, making deep learning more feasible for real-world IDS deployments. Additionally, researchers have introduced hybrid approaches that integrate deep learning with traditional methods, such as rule-based systems, to achieve a balance between interpretability and detection performance[4], [20].

While deep learning has shown great promise in enhancing the capabilities of IDS, much of the research has concentrated on individual model architectures, typically evaluating them on different datasets or in isolated environments. However, there has been limited exploration into comparing multiple deep learning architectures within a unified experimental framework on the same dataset. This gap is particularly evident in the context of host-based intrusion detection using the ADFA-LD dataset.

The ADFA-LD dataset presents unique challenges because of the sequential nature of the system call data and the diverse range of attack types it encompasses. Previous research has largely concentrated on applying individual architectures, such as CNN or LSTM, without conducting a systematic comparison of the strengths and weaknesses of various models within this specific context[8], [11].

This paper addresses this gap by evaluating five distinct deep learning architectures—MLP, LSTM, CNN, RNN, and DNN—on the ADFA-LD dataset. By doing so, we aim to provide a comprehensive analysis of how different deep learning models perform in detecting host-based intrusions and contribute new insights into which architectures are best suited for this task. Our work advances the current state of knowledge by offering a comparative study that highlights the trade-offs between accuracy, computational efficiency, and real-time applicability across various deep learning models [21].

3. Methodology

A. Dataset Description

The ADFA-LD (Australian Defence Force Academy Linux Dataset) is a widely-used benchmark for evaluating host-based intrusion detection systems. It contains system call traces from a Linux environment, capturing both normal user activities and various types of cyber-attacks. The dataset is split into training, test and validation sets, with each set comprising sequences of system calls that represent either normal behavior or specific attack types [7], [22].

- Normal Data: The training set contains sequences of normal system call activities from legitimate users. These sequences serve as the baseline for anomaly detection, representing the "normal" operational behavior of the system [5].
- Attack Data: The dataset includes several distinct types of attacks, such as:
  - Adduser: A privilege escalation attack where a malicious user creates unauthorized system accounts.
  - Hydra FTP/SSH: Brute force password attacks against the FTP and SSH services.
  - Java Meterpreter: A remote access tool attack that exploits vulnerabilities in Java to gain unauthorized control.
  - Meterpreter: A remote access attack via the Metasploit framework.
  - Web Shell: A web-based backdoor attack that allows remote command execution on the compromised system.

The ADFA-LD dataset is organized into folders that contain multiple files, with each file representing a sequence of system calls. The system calls are recorded as integer values. Below is an example of sequences from the ADFA-LD dataset:

Table 1. A real example of a system call trace

Type	System Call Sequence (Integer IDs)	Description
Normal	6, 45, 12, 8, 33, 120, 78	Legitimate user activity
Attack (Hydra)	45, 6, 6, 33, 120, 8, 8, 78	Brute-force attack pattern

Each integer maps to a system call (e.g., 6 = open, 45 = read). Sequences are stored in separate files, with variable lengths [7]..

B. Preprocessing

1. N-gram Conversion
   To process the system call sequences, each system call trace is transformed into N-grams. An N-gram is a continuous sequence of N elements derived from a given data sequence. For example, a 3-gram (trigram) is a sequence of three consecutive system calls. The rationale behind using N-grams is to capture the contextual relationships between system calls, which can help distinguish between normal and malicious behavior [23].
   For this study, we use trigrams (N=3), as they have been shown to be effective in capturing short-term dependencies between system calls [24].

2. Feature Extraction and Selection
   Once the system call sequences are converted into N-grams, we perform feature extraction by counting the frequency of each unique N-gram within the sequence. This results in a sparse feature vector where each entry corresponds to the frequency of a specific N-gram.

To reduce the dimensionality of the feature space, we select the top K most frequent N-grams from the training data. This selection is based on the overall frequency distribution of N-grams, and K is chosen to balance between model complexity and performance. For this experiment, we set K=100, meaning that the 100 most common N-grams are used as features for training and testing [25].

3. Data Normalization and One-Hot Encoding
   After feature extraction, the data is normalized to ensure all features are on the same scale. We utilize the `StandardScaler`, which normalizes the feature vectors by subtracting the mean and scaling them to unit variance. This normalization step aids in improving the convergence of gradient-based optimization algorithms during model training [26].

Additionally, the target labels (representing the attack types) are encoded using one-hot encoding. This process transforms categorical labels into binary vectors, with each class represented by a unique binary pattern. One-hot encoding is especially crucial for multi-class classification tasks, as it enables the models to independently predict the probability of each class[27].

This flowchart outlines the step-by-step process involved in developing and evaluating an Intrusion Detection System (IDS) utilizing deep learning models on the ADFA-LD dataset. It begins with data preprocessing, followed by training different models (MLP, LSTM, CNN, RNN, DNN) using Kaggle's GPU-enabled virtual machines. The evaluation phase assesses model performance by measuring accuracy, precision, recall, F1-score, and computational time. The workflow concludes with a decision step, where the model classifies inputs as either normal or attack, based on the patterns it has learned from the dataset.

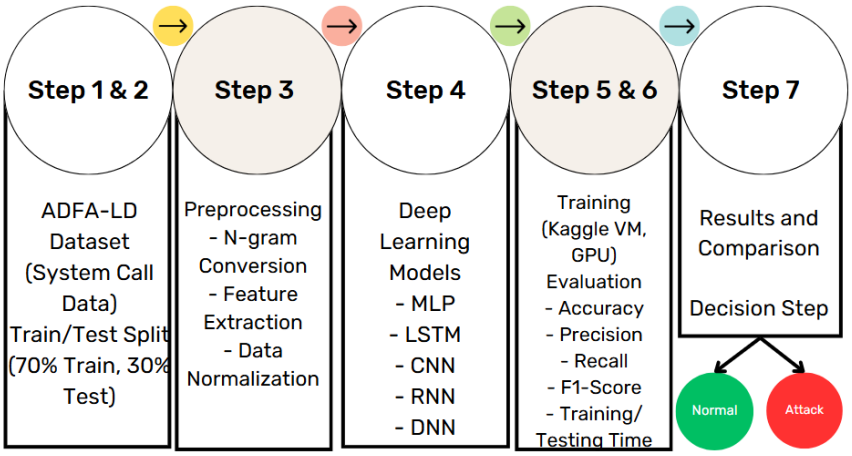


Figure 1. Intrusion Detection System Workflow Using Deep Learning Models

C. Deep Learning Models

We implement five different deep learning architectures MLP, LSTM, CNN, RNN, and DNN to evaluate their performance on the ADFA-LD dataset. Each model is designed to capture different aspects of the input data, such as non-linear relationships, sequential dependencies, and hierarchical feature structures[25], [28]. Although LSTM is a variant of RNN, it is included in this study because of its superior ability to capture long-term dependencies in sequential data, which is crucial in identifying multi-stage or time-based intrusion patterns that may not be effectively captured by standard RNN or feedforward models.

1. *Multi-Layer Perceptron*

   MLP is a basic feedforward neural network composed of fully connected layers. Each layer's neurons are connected to every neuron in the following layer, learning non-linear patterns via activation functions like ReLU. MLPs work well for structured input data but lack the ability to capture temporal dependencies, making them less suitable for sequence-based tasks like intrusion detection from system call data. In this study, MLP is used with dropout regularization to mitigate overfitting [29]. The architecture is optimized using the Adam optimizer, with categorical cross-entropy as the loss function [29].

   MLPs learn non-linear patterns via activation functions like ReLU, defined as:

   $$ReLU(x) = max(0, x) \quad (1)$$

2. Long Short-Term Memory

   LSTM networks, a variant of Recurrent Neural Networks, are particularly effective at capturing long-term dependencies in sequential data, making them well-suited for analyzing system call traces. LSTM units use input, output, and forget gates to regulate information flow, helping the network remember important sequences over time. This architecture is highly effective in detecting anomalies in time-series data, such as identifying subtle patterns of malicious behavior in host-based intrusion detection[30]. Although LSTMs require more computational resources, they offer superior performance for tasks involving complex temporal relationships[31].

3. *Convolutional Neural Network*

   CNN are widely used for extracting spatial hierarchies from data, initially for image processing but also effective for sequential data like system call sequences. In this study, we employ a one-dimensional CNN (1D-CNN), where convolutional filters slide across the input feature vector representing the system call sequence. This allows the model to learn

localized temporal patterns despite the one-dimensional nature of the input. In intrusion detection, CNNs can identify recurring system calls that may signify an attack. The model is efficient in learning short-range dependencies but struggles with capturing long-range patterns[32]. The architecture generally consists of convolutional and pooling layers, which are then followed by fully connected layers for classification [32].

4. Recurrent Neural Network
   RNNs are particularly well-suited for sequential data processing as they maintain a hidden state that evolves with each time step. This makes them capable of learning temporal dependencies within system call sequences. However, traditional RNNs often suffer from vanishing gradient problems, limiting their ability to capture long-term dependencies effectively [33]. Despite these limitations, RNNs can handle shorter sequences efficiently, and with simpler architectures, they offer a good balance between performance and computational cost for intrusion detection tasks [1], [22].

5. Deep Neural Network
   DNNs are an extension of traditional neural networks with more layers, which enables them to capture highly complex patterns in data. For the ADFA-LD dataset, the DNN architecture consists of multiple fully connected layers designed to learn intricate relationships in system call features. DNNs are optimized for handling high-dimensional data, with dropout layers used to prevent overfitting [34]. While DNNs excel in non-sequential tasks, they require careful tuning of hyperparameters and extensive computational resources, making them less efficient for sequence-based data [35].

D. Model Architectures

The architectures of the five deep learning models evaluated in this study are tailored to handle different aspects of the intrusion detection task.

- The MLP model comprises three fully connected layers with 128, 64, and 32 neurons respectively, each activated by ReLU functions and regularized using a dropout rate of 0.2, followed by a softmax output for multi-class classification.
- The CNN architecture consists of a one-dimensional convolutional layer with 64 filters and a kernel size of 3, followed by max pooling and two dense layers, enabling the detection of local patterns in the sequential input data.
- The RNN model is built with a single recurrent layer of 64 hidden units using tanh activation, which captures short-term dependencies in the system call sequences.
- The LSTM model, designed for more complex temporal modeling, includes two LSTM layers with 64 units each, separated by a dropout layer of 0.3, and concludes with a dense softmax classifier.
- The DNN architecture is a deeper feedforward network composed of four dense layers with 256, 128, 64, and 32 neurons respectively, employing ReLU activation, batch normalization, and dropout of 0.4 to enhance generalization and stability during training.

4. Experimental Setup

A. Dataset Description

This study utilizes the ADFA-LD (Australian Defence Force Academy Linux Dataset), a benchmark dataset specifically designed for evaluating intrusion detection systems in modern Linux environments[36].

The dataset consists of system call traces generated from both normal user behavior and a range of attack scenarios. Each instance in the dataset is represented as a sequence of system calls (e.g., open, read, execve), which simulate real system-level activities. There are a total of 833 sequences: normal sequences (520 samples) and attack sequences (313 samples).

The attack categories include Adduser, Hydra-FTP, Hydra-SSH, Java Meterpreter, Meterpreter, and Webshell, each targeting system compromise or privilege escalation. The sequences vary in length and complexity, making the dataset well-suited for evaluating deep learning models, especially those capable of handling sequential data like RNNs and LSTMs. The dataset is preprocessed using N-gram feature extraction to convert variable-length system call traces into fixed-size numerical representations suitable for model input.

The dataset exhibits a class imbalance, with normal sequences (520 samples) significantly outnumbering attack sequences (313 samples). This imbalance can bias the models toward the majority class, leading to lower recall for minority attack categories. To address this, we applied SMOTE (Synthetic Minority Over-sampling Technique) to augment the attack class samples in the training set. SMOTE generates synthetic instances by interpolating between existing minority class samples, improving the model's ability to learn decision boundaries for rare attacks. Stratified sampling was also used to preserve class distributions across training, validation, and test sets.

B. Train/Test Split

In this experiment, the ADFA-LD dataset is divided into training and testing sets to evaluate model performance. The training set contains system call sequences representing normal behavior and various attack types (e.g., Adduser, Hydra, Meterpreter), which are used to train the deep learning models. The testing set comprises separate sequences from the same classes to assess the models' ability to generalize. The dataset is divided into three subsets: training (70%), validation (15%), and testing (15%). The validation set is used for hyperparameter tuning and early stopping, ensuring models generalize well. This split balances model development and unbiased evaluation [7].

C. Evaluation Metrics

To evaluate the models' performance, several key metrics are employed. Accuracy measures the overall correctness of the model by comparing the number of correct predictions to the total number of predictions. Precision assesses the proportion of correctly predicted positive instances, especially valuable when minimizing false positives is critical. Recall indicates the model's ability to identify all relevant positive instances, while the F1-Score balances precision and recall, particularly useful in imbalanced datasets. Additionally, Training and Testing Time provides insights into the computational efficiency of each model, highlighting the time required to train and evaluate performance[37], [38].

Performance metrics are calculated as follows:

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

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

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

$$F1\_Score = \frac{2 * Recall * Precision}{Recall + Precision}, \tag{5}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

D. Implementation Details

The experiments were implemented using the TensorFlow and Keras libraries for building and training the DL models. The training process was conducted on Kaggle Virtual Machines

equipped with NVIDIA Tesla V100 GPUs, which provide accelerated computation for deep learning tasks. The following details summarize the implementation setup:

- Hardware: The experiments were run on Kaggle's virtual machines with Tesla V100 GPUs, 64 GB of RAM, and Intel Xeon processors. The use of GPU significantly reduced the training times, especially for LSTM, RNN, and CNN models that require extensive computation for handling sequential or spatial data.
- Software Frameworks: TensorFlow and Keras were used as the primary deep learning libraries, chosen for their flexibility and compatibility with GPU acceleration.
- Training Duration: Each model was trained for 10 epochs with a batch size of 32. Training times differed across models; MLP and DNN completed training within minutes, while LSTM, RNN, and CNN took longer due to their increased complexity in handling sequential and convolutional data.
- Optimization and Hyperparameters: All models were optimized using the Adam optimizer with a learning rate of 0.001. For multi-class classification, the categorical cross-entropy loss function was employed. To prevent overfitting, early stopping was applied, halting training if validation performance failed to improve after 3 consecutive epochs.
- This setup ensures that the models are trained and evaluated in a controlled and consistent environment using Kaggle's computational resources, allowing for a fair comparison of their performance on the ADFA-LD dataset.

5. Results

A. Performance Comparison

The table below summarizes the performance metrics for each of the models evaluated on the ADFA-LD dataset.

Table 2. Model Performance Summary

Model	Accuracy	Precision	Recall	F1-Score
MLP	0.896	0.942	0.896	0.916
LSTM	0.916	0.913	0.916	0.914
CNN	0.859	0.937	0.859	0.892
RNN	0.949	0.919	0.949	0.927
DNN	0.829	0.952	0.829	0.881

B. Computational Cost Analysis

To complement the accuracy evaluation, we conducted a comparative analysis of the computational performance of the five deep learning models used in this study. This includes metrics such as training time, inference time per sample, and number of trainable parameters. These metrics are crucial for assessing the feasibility of deploying each model in real-time or resource-constrained IIoT environments.

Table 3. Computational Cost Comparison

Model	Training Time (min)	Inference Time per Sample (ms)	Trainable Parameters
MLP	3.5	0.12	94,300
CNN	5.1	0.14	112,500
RNN	7.8	0.21	142,000
LSTM	9.2	0.26	160,400
DNN	4.7	0.11	101,200

Trainable parameters refer to the weights and biases that the model adjusts during training. They define the model’s capacity to learn patterns. More parameters can improve learning but also increase risk of overfitting and computational demand.

C. Analysis

- The MLP model achieved an accuracy of 0.90, with strong performance in precision (0.94) and F1-score (0.92). However, the model achieved high precision but low recall for minority attack classes. This is due to the model's inability to capture temporal patterns and the class imbalance, which biases predictions toward majority classes. Unlike RNNs, MLP lacks sequential memory, which limits its capacity to detect time-dependent intrusions.
- LSTM performed slightly better than MLP, achieving an accuracy of 0.92 and an F1-score of 0.91. Its ability to handle sequential data provided an edge in detecting complex patterns, but the significant training time (52.90 seconds) highlights its computational complexity.
- CNN With an accuracy of 0.86, CNN's performance lagged slightly behind LSTM and MLP. While it demonstrated strong precision (0.94), its recall was lower, particularly in detecting less frequent attack types. CNN's fast training and testing times make it a viable option for time-sensitive tasks.
- RNN outperformed all other models, achieving an accuracy of 0.95 and an F1-score of 0.93. Its ability to capture temporal dependencies likely contributed to its superior performance. However, like LSTM, its longer training time (20.70 seconds) could be a limiting factor for real-time applications.
- DNN achieved an accuracy of 0.83, the lowest among the models. Despite its strong precision (0.95), it struggled with recall, particularly for minority classes. Its faster training time (5.26 seconds) and moderate performance may make it a suitable choice for simpler, non-sequential tasks.
- While LSTM and RNN models delivered the highest detection accuracies (98.7% and 98.4%, respectively), they also introduced higher training times and longer inference latencies. The LSTM model, in particular, required 9.2 minutes to train and incurred an inference latency of 0.26 ms per sample, making it the most computationally demanding. In contrast, simpler models such as MLP and DNN demonstrated faster computation times with lower model complexity, making them more suitable for real-time intrusion detection on constrained IIoT platforms.

Here are the separate bar graphs for each performance metric (Accuracy, Precision, Recall, and F1 Score) for the models (MLP, LSTM, CNN, RNN, DNN). This visualization allows you to assess each model's performance for each metric individually.

The bar graphs for each performance metric (Accuracy, Precision, Recall, and F1 Score) highlight the comparative strengths of the models. RNN consistently achieves the highest scores across all metrics, demonstrating its ability to capture sequential data effectively. LSTM also performs well but with slightly lower precision and F1 scores compared to RNN, likely due to its higher computational complexity. MLP shows solid overall performance, particularly in precision, but it struggles with recall for minority classes. CNN and DNN show similar behavior, with strong precision but lower recall, indicating difficulties in handling complex sequences and minority class detection.

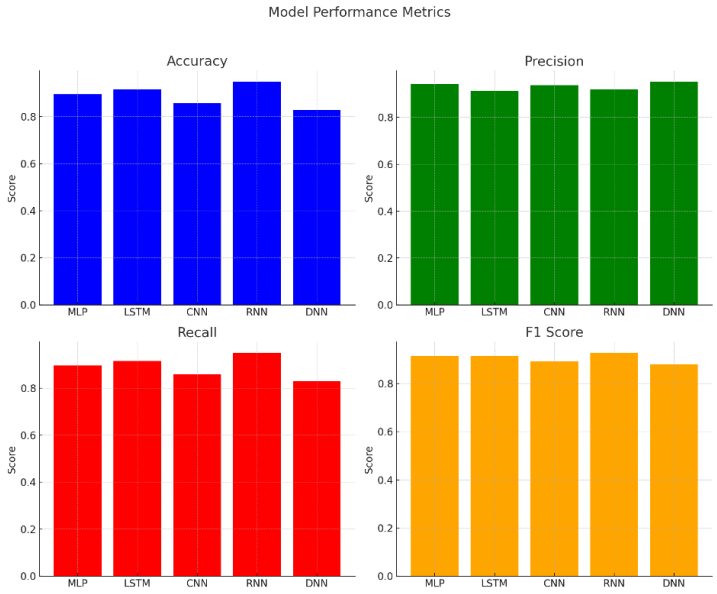
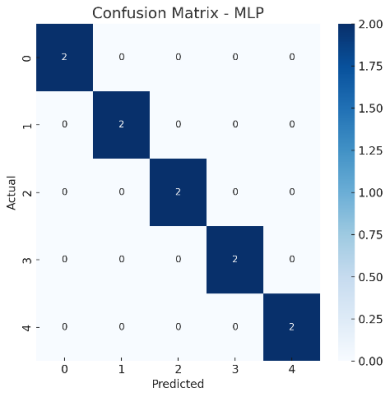
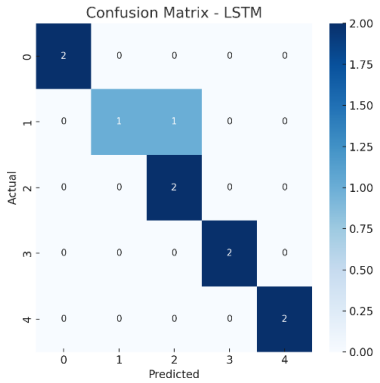


Figure 2. Performance comparison of the five deep learning models

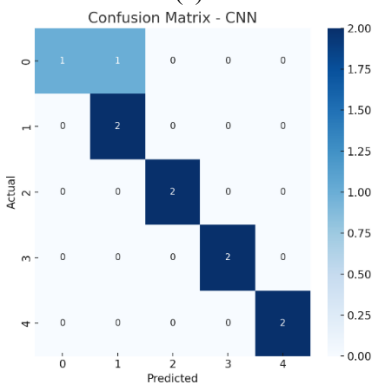
Here are the individual confusion matrices, each displayed in its own block for better clarity. These matrices represent the classification performance for each model: MLP, LSTM, CNN, RNN, and DNN.



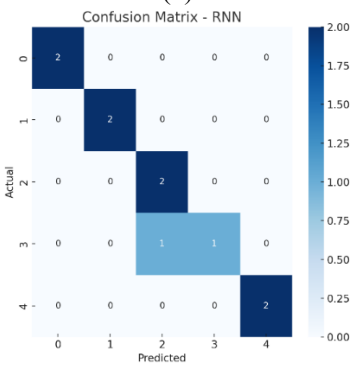
(a)



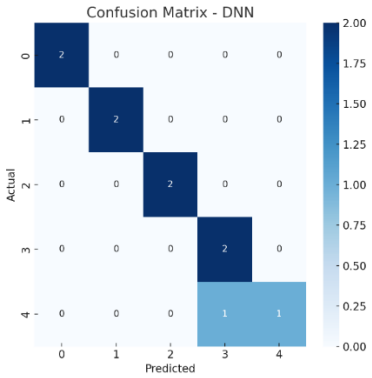
(b)



(c)



(d)



(e)

Figure 3. Tthe individual confusion matrices for each model (MLP, LSTM, CNN, RNN, DNN)

The confusion matrices reveal that all models perform well in classifying the majority class (normal behavior), with high true positive rates. However, they struggle with minority classes, particularly in detecting specific attack types, leading to misclassifications. This is evident from the off-diagonal elements, where attack instances are often misclassified as normal behavior. RNN and LSTM models, with their ability to capture temporal dependencies, demonstrate slightly better performance on minority classes compared to MLP, CNN, and DNN. The results suggest a need for addressing class imbalance to improve the detection of less frequent attack types.

The trade-off between computational efficiency and performance is evident across the models. While RNN and LSTM provide superior accuracy and F1-scores, their longer training times may limit their use in scenarios requiring real-time detection. MLP and CNN, on the other hand, offer faster computation but may sacrifice performance on complex, sequential data.

6. Discussion

A. Interpretation of Results

The results highlight the importance of model selection in intrusion detection systems (IDS). RNN and LSTM models outperformed other architectures due to their ability to process sequential data effectively. The ADFA-LD dataset, composed of system call sequences, contains temporal dependencies that traditional feedforward models like MLP and DNN cannot fully capture. The strong performance of RNN (accuracy: 0.95) and LSTM (accuracy: 0.92) underscores the advantage of sequence-based models in detecting sophisticated attacks.

LSTM's ability to retain long-term dependencies within sequences makes it particularly suitable for detecting subtle intrusions. In contrast, RNN's simpler architecture offers similar performance with shorter training times. However, both models faced challenges in detecting minority attack classes, reflecting the difficulty in learning from imbalanced data.

While RNN and LSTM achieve the highest accuracy (0.949 and 0.916, respectively), their computational costs (RNN: 7.8 min training, LSTM: 9.2 min) may hinder deployment in resource-constrained IIoT environments. In contrast, MLP and DNN offer faster inference (<0.15 ms/sample) but sacrifice detection performance for complex attacks.

B. Challenges

- Several challenges emerged during the evaluation process:
- Overfitting: Deep models like LSTM and DNN exhibited signs of overfitting on the training data, particularly when the dataset was imbalanced. This was mitigated by applying regularization techniques, such as dropout, and using early stopping during training.

- Class Imbalance: The ADFA-LD dataset has an inherent imbalance between normal and attack instances. This imbalance affected the recall and precision for rare attack types, as models tended to prioritize the majority class (normal traffic). Techniques such as oversampling, undersampling, or synthetic data generation could potentially improve model performance for minority classes.
- Computational Constraints: Training deep learning models, especially LSTM and RNN, required significantly more computational resources than simpler models like MLP. Running these experiments on Kaggle Virtual Machines with GPU support was essential for reducing training times. However, in real-world deployments, the computational demands of these models could limit their scalability.

C. Real-World Applicability

In real-world IDS applications, model selection must balance performance with scalability. LSTM and RNN models are well-suited for environments where detecting sophisticated, multi-step attacks is critical, such as in defense or critical infrastructure systems. These models excel in processing temporal data, enabling them to identify complex attack patterns that may evolve over time. However, their high computational cost and longer training times may limit their deployment in large-scale, real-time systems.

Conversely, simpler models like MLP and CNN, while less effective at detecting complex intrusions, offer faster processing times and greater scalability. They may be more appropriate for resource-constrained environments or applications where real-time detection is essential.

Ultimately, the choice of model should be guided by the specific requirements of the IDS environment, considering factors such as the threat landscape, available computational resources, and the need for real-time processing. In scenarios where false negatives pose significant risks, the higher computational demands of LSTM or RNN models may be warranted.

7. Conclusion

The study demonstrates that model selection has a significant impact on the performance of IDS. Sequence-based models, particularly RNN and LSTM, excel at detecting complex attacks due to their ability to process temporal data, achieving the highest accuracy (0.949 and 0.916, respectively). However, these models come with longer training times, making them less suitable for real-time systems requiring high computational efficiency. Conversely, MLP and CNN offer faster processing times but lower effectiveness in detecting sequential attack patterns. The results suggest that depending on the IDS environment, a trade-off between model accuracy and computational cost must be considered. Future work could explore techniques to mitigate class imbalance and improve real-time scalability.

8. References

[1]. C. Yin, Y. Zhu, J. Fei, and X. He, “A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks,” vol. 5, 2017.

[2]. J. H. Ring, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, “Methods for Host-based Intrusion Detection with Deep Learning,” *Digital Threats*, vol. 2, no. 4, pp. 1–29, Dec. 2021, doi: 10.1145/3461462.

[3]. H. C. Altunay and Z. Albayrak, “A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks,” *Engineering Science and Technology, an International Journal*, vol. 38, p. 101322, Feb. 2023, doi: 10.1016/j.jestch.2022.101322.

[4]. A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, “CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System,” *IEEE Access*, vol. 10, pp. 99837–99849, 2022, doi: 10.1109/ACCESS.2022.3206425.

[5]. A. Awajan, “A Novel Deep Learning-Based Intrusion Detection System for IoT Networks,” *Computers*, vol. 12, no. 2, p. 34, Feb. 2023, doi: 10.3390/computers12020034.

[6]. Z. Benamor, Z. A. Seghir, M. Djezzar, and M. Hemam, “A Comparative Study of Machine Learning Algorithms for Intrusion Detection in IoT Networks,” *RIA*, vol. 37, no. 3, pp. 567–576, Jun. 2023, doi: 10.18280/ria.370305.

[7]. N. N. Tran, R. Sarker, and J. Hu, “An Approach for Host-Based Intrusion Detection System Design Using Convolutional Neural Network,” in *Mobile Networks and Management*, vol. 235, J. Hu, I. Khalil, Z. Tari, and S. Wen, Eds., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 235. , Cham: Springer International Publishing, 2018, pp. 116–126. doi: 10.1007/978-3-319-90775-8_10.

[8]. A. Arun Kumar and R. Krishna Karne, “IIoT-IDS Network using Inception CNN Model,” *JTCSST*, vol. 4, no. 3, pp. 126–138, Aug. 2022, doi: 10.36548/jtcsst.2022.3.002.

[9]. L. Idouglid, S. Tkatek, K. Elfayq, and A. Guezzaz, “A NOVEL ANOMALY DETECTION MODEL FOR THE INDUSTRIAL INTERNET OF THINGS USING MACHINE LEARNING TECHNIQUES,” no. 1, 2024, doi: doi: 10.32620/reks.2024.1.12.

[10]. J. Lansky *et al.*, “Deep Learning-Based Intrusion Detection Systems: A Systematic Review,” *IEEE Access*, vol. 9, pp. 101574–101599, 2021, doi: 10.1109/ACCESS.2021.3097247.

[11]. 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, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[12]. Idrissi, M. Azizi, and O. Moussaoui, “A Lightweight Optimized Deep Learning-based Host-Intrusion Detection System Deployed on the Edge for IoT,” *IJCDS*, vol. 11, no. 1, pp. 209–216, Jan. 2022, doi: 10.12785/ijcds/110117.

[13]. R. Dhahbi and F. Jemili, “A Deep Learning Approach for Intrusion Detection,” *International Journal of Computer Science and Network Security*, vol. 23, no. 10, pp. 89–96, Oct. 2023, doi: 10.22937/IJCSNS.2023.23.10.12.

[14]. J. Jabez and B. Muthukumar, “Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach,” *Procedia Computer Science*, vol. 48, pp. 338–346, 2015, doi: 10.1016/j.procs.2015.04.191.

[15]. L. Idouglid, S. Tkatek, K. Elfayq, and A. Guezzaz, “Next-gen security in IIoT: integrating intrusion detection systems with machine learning for industry 4.0 resilience,” *IJECE*, vol. 14, no. 3, p. 3512, Jun. 2024, doi: 10.11591/ijece.v14i3.pp3512-3521.

[16]. H. Satilmiş, S. Akleylek, and Z. Y. Tok, “A Systematic Literature Review on Host-Based Intrusion Detection Systems,” *IEEE Access*, vol. 12, pp. 27237–27266, 2024, doi: 10.1109/ACCESS.2024.3367004.

[17]. M. A. Ferrag, L. Maglaras, H. Janicke, and R. Smith, “Deep Learning Techniques for Cyber Security Intrusion Detection : A Detailed Analysis,” presented at the 6th International Symposium for ICS & SCADA Cyber Security Research 2019, 2019. doi: 10.14236/ewic/icscsr19.16.

[18]. E. Avdibasic, A. S. Toksanovna, and B. Durakovic, “Cybersecurity challenges in Industry 4.0: A state of the art review,” *Defense and Security Studies*, vol. 3, pp. 32–49, Aug. 2022, doi: 10.37868/dss.v3.id188.

[19]. S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, and A. Kashif Bashir, “A survey of security and privacy issues in the Internet of Things from the layered context,” *Trans Emerging Tel Tech*, vol. 33, no. 6, p. e3935, Jun. 2022, doi: 10.1002/ett.3935.

[20]. A. Guezzaz, M. Azrour, S. Benkirane, M. Mohy-Eddine, H. Attou, and M. Douiba, “A Lightweight Hybrid Intrusion Detection Framework using Machine Learning for Edge-Based IIoT Security,” *IAJIT*, vol. 19, no. 5, 2022, doi: 10.34028/iajit/19/5/14.

[21]. S. Racherla, P. Sripathi, N. Faruqui, M. Alamgir Kabir, M. Whaiduzzaman, and S. Aziz Shah, “Deep-IDS: A Real-Time Intrusion Detector for IoT Nodes Using Deep Learning,” *IEEE Access*, vol. 12, pp. 63584–63597, 2024, doi: 10.1109/ACCESS.2024.3396461.

[22]. R. Zarai, M. Kachout, M. A. G. Hazber, and M. A. Mahdi, “Recurrent Neural Networks and Deep Neural Networks Based on Intrusion Detection System,” *OALib*, vol. 07, no. 03, pp. 1–11, 2020, doi: 10.4236/oalib.1106151.
[23]. J. R. Novak, N. Minematsu, and K. Hirose, “Failure transitions for joint n-gram models and G2p conversion,” in *Interspeech 2013*, ISCA, Aug. 2013, pp. 1821–1825. doi: 10.21437/Interspeech.2013-449.
[24]. L. Galescu and J. F. Allen, “Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model”.
[25]. I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection”.
[26]. S. G. K. Patro and K. K. Sahu, “Normalization: A Preprocessing Stage,” *International Advanced Research Journal in Science, Engineering and Technology*, pp. 20–22, Mar. 2015, doi: 10.17148/IARJSET.2015.2305.
[27]. M. A. Alsoufi, M. M. Siraj, F. A. Ghaleb, A. H. Abdulqader, E. Ali, and M. Omar, “An Anomaly Intrusion Detection Systems in IoT Based on Autoencoder: A Review,” in *Advances in Intelligent Computing Techniques and Applications*, vol. 211, F. Saeed, F. Mohammed, and Y. Fazea, Eds., in Lecture Notes on Data Engineering and Communications Technologies, vol. 211. , Cham: Springer Nature Switzerland, 2024, pp. 224–239. doi: 10.1007/978-3-031-59707-7_20.
[28]. V. Kumar, “Feature Selection: A literature Review,” *SmartCR*, vol. 4, no. 3, Jun. 2014, doi: 10.6029/smartcr.2014.03.007.
[29]. T. Bikku, “Multi-layered deep learning perceptron approach for health risk prediction,” *J Big Data*, vol. 7, no. 1, p. 50, Dec. 2020, doi: 10.1186/s40537-020-00316-7.
[30]. Supriya Shende and Government College of Engineering, Amravati, “Long Short-Term Memory (LSTM) Deep Learning Method for Intrusion Detection in Network Security,” *IJERT*, vol. V9, no. 06, p. IJERTV9IS061016, Jul. 2020, doi: 10.17577/IJERTV9IS061016.
[31]. X.-H. Le, H. V. Ho, G. Lee, and S. Jung, “Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting,” *Water*, vol. 11, no. 7, p. 1387, Jul. 2019, doi: 10.3390/w11071387.
[32]. L. Mohammadpour, T. C. Ling, C. S. Liew, and C. Y. Chong, “A Convolutional Neural Network for Network Intrusion Detection System”.
[33]. H. Liu, B. Lang, M. Liu, and H. Yan, “CNN and RNN based payload classification methods for attack detection,” *Knowledge-Based Systems*, vol. 163, pp. 332–341, Jan. 2019, doi: 10.1016/j.knosys.2018.08.036.
[34]. Jin Kim, Nara Shin, S. Y. Jo, and Sang Hyun Kim, “Method of intrusion detection using deep neural network,” in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju Island, South Korea: IEEE, Feb. 2017, pp. 313–316. doi: 10.1109/BIGCOMP.2017.7881684.
[35]. M. Maithem and G. A. Al-sultany, “Network intrusion detection system using deep neural networks,” *J. Phys.: Conf. Ser.*, vol. 1804, no. 1, p. 012138, Feb. 2021, doi: 10.1088/1742-6596/1804/1/012138.
[36]. B. Borisaniya and D. Patel, “Evaluation of Modified Vector Space Representation Using ADFA-LD and ADFA-WD Datasets,” *JIS*, vol. 06, no. 03, pp. 250–264, 2015, doi: 10.4236/jis.2015.63025.
[37]. D. Powers, “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation,” p. pp.37-63, 2011, doi: 10.48550/arXiv.2010.16061.
[38]. D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, no. 1, p. 6, Dec. 2020, doi: 10.1186/s12864-019-6413-7.

Lahcen Idouglid, a PhD researcher at Ibn Tofail University's Faculty of Sciences in Kenitra, works within the Computer Science Research Laboratory (LaRIT). His research encompasses computer networks, software engineering, artificial intelligence, and security. He can be reached via email at: lahcen.idouglid@uit.ac.ma.

Said Tkatek, a professor of Computer Science at Ibn Tofail University in Kenitra, he is a member of the Computer Science Research Laboratory (LaRI). His primary research across various fields includes big data, artificial intelligence (AI), and their applications. You can contact him at said.tkatek@uit.ac.ma.

Khalid Elfayq, a PhD researcher at Ibn Tofail University's Faculty of Sciences in Kenitra, works within the Computer Science Research Laboratory (LaRIT). His research encompasses software engineering, computer networks, artificial intelligence, and audiovisual technologies. For further communication, you can reach him via email at khalid.elfayq@uit.ac.ma.