Scientific Research Publishing

Journal of Computer and Communications, 2025, 13(9), 77-103
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Improving the Performance of a Data-Driven Intelligent Model for Intrusion Detection in IoT Networks

Severin Nguendap[1], Hamdane Allamine Moussa[1], Eric Fotsing[2], Armand Nzeukou[1], Jean Pierre Lienou[1]

[1]Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon
[2]Department of Computer Engineering, IUT-FV, University of Dschang, Bandjoun, Cameroon
Email: snguendap@gmail.com, efotsing@univ-dschang.org, armand.nzeukou@univ-dschang.org, jp.lienou@univ-dschang.org

How to cite this paper: Nguendap, S., Moussa, H.A., Fotsing, E., Nzeukou, A. and Lienou, J.P. (2025) Improving the Performance of a Data-Driven Intelligent Model for Intrusion Detection in IoT Networks. *Journal of Computer and Communications*, **13**, 77-103.
https://doi.org/10.4236/jcc.2025.139005

Received: August 21, 2025
Accepted: September 23, 2025
Published: September 26, 2025

Abstract

This work contributes to the development of intelligent data-driven approaches to improve intrusion management in smart IoT environments. The proposed model combines a hybrid AutoEncoder-LSTM designed to capture the temporal structure of network traffic and transform latent representations into explicit predictions. An analysis of the model results on two datasets, NSL-KDD and IoT-23, and aims to improve intrusion detection in IoT environments. The NSL-KDD dataset comprises data of 41 attributes, including 38 continuous and 3 discrete, which are pre-processed by cleaning (outlier removal), one-hot encoding of categorical variables (protocol_type, service, flag), conversion to numeric format, normalization, and rebalancing (50% Normal, 22% DoS, 13% Probe, 12% R2L, 3% U2R), with anomaly detection based on MSELoss, achieving 98.88% accuracy, precision, recall, and F1-Score. For the IoT-23 dataset, which includes data from 20 malware captures and 3 harmless traffic, the dataset undergoes preprocessing including removal of unlabeled or duplicate lines, conversion to numerical values, dimensionality reduction (correlation > 0.95), and correlation heatmap, with an unbalanced distribution (56.6% Normal, 21.3% Infiltration, 18.9% Gagfyt, 3.2% Mirai Bruteforce, 0.1% Mirai DDoS), providing 99.02% accuracy, 99.05% precision, 99.07% recall, and 99.09% F1-Score. Both models demonstrate rapid convergence and good generalization, but the AE-LSTM model applied to the IoT-23 dataset performs better thanks to its distribution favoring normal reconstruction, slightly outperforming the NSL-KDD model despite the diversity of attacks covered by the latter. Our model is suitable for intrusion detection in smart IoT environments, combining robustness and generalization capacity.

Keywords

IoT Environments, Intrusion Detection, AE-LSTM, IoT-23, NSL-KDD

1. Introduction

The Internet of Things (IoT) has established itself as an essential technology in several fields such as healthcare, smart cities, industry, and agriculture. The implementation of this recent technology, which is growing daily, makes it possible to interconnect all the objects near us. This rapid growth is leading to a proliferation of connected objects, often heterogeneous, with limited resources, and poorly protected. The security of IoT environments is now a major, even critical, concern, especially since traditional detection methods, based on signatures or static rules, struggle to identify new, evolving, or complex attacks. Thus, it is necessary to develop more efficient and more suitable models to detect malicious activity as quickly and accurately as possible. To overcome these limitations, much research is now relying on artificial intelligence techniques to design intrusion management systems (IMS) capable of learning, adapting, and detecting abnormal behavior in smart IoT environments. An effective IMS must rely on an effective and diverse dataset. We propose a hybrid IMS using autoencoders (AE) and long-term memory (LSTM) neural networks to intelligently detect cyberattacks. AE-LSTM models allow us to both extract the internal features of sequences and capture their temporal dynamics.

In our work, we perform a comparative analysis of the results of the model trained on two datasets, NSL-KDD and IoT-23, to assess the model's performance in providing clear, accurate, and understandable detections.

This work is part of this dynamic, and therefore, the main objective is to analyze the model's performance on these datasets.

Two specific objectives guide this study:

1) Design a robust, hybrid IMS model adapted to IoT environments to efficiently capture network traffic time sequences and abnormal behaviors.

2) Analyze the model's performance applied to the NSL-KDD and IoT-23 datasets.

This article is organized as follows: Section II provides a literature review that provides an overview of IMS approaches in the IoT. Section III is devoted to the proposed methodology. Section IV presents the results and discussion, followed by Section V, a conclusion and outlook.

2. State of the Art

This section provides an overview of intrusion detection approaches in the IoT, emphasizing the main concerns related to IoT environments, the limitations of traditional systems, and the contributions of intelligent systems. It also proposes

the advantages of intelligent data-driven approaches. Smart IoT environments include a wide and diverse range of resource-constrained devices and varying communication standards. This heterogeneity complicates traditional security mechanisms. IoT architectures often follow layered models, each of which presents distinct vulnerabilities. Many IoT communication protocols, such as MQTT, CoAP, AMQP, and DDS, were designed to be lightweight with robust security in mind. The IMS must be capable of detecting and responding in real time.

Mahmoud *et al.* [1] propose an AE-LSTM model designed to detect intrusions in IoT networks, based on the NSL-KDD dataset. The model is structured around a six-layer deep autoencoder, combined with LSTM layers to capture the temporal structure of network flows. To improve the learning reliability, the authors introduced a preprocessing method aimed at eliminating outliers, which are often the cause of data imbalance. The model was evaluated in both binary (normal vs. malicious traffic) and multiclass (DoS, Probe, R2L, U2R, Normal) classification. The results show an accuracy of 98.88% and an equivalent F1-Score, confirming the robustness of this architecture for flow analysis in control environments.

Cosimo *et al.* [2] combined a convolutional neural network (CNN) with a bidirectional LSTM (BiLSTM), along with sampling techniques, to address data imbalance in datasets such as NSL-KDD and UNSW-NB15. They achieved the following performance: Binary accuracy: 98.27% on NSL-KDD and 99.87% on UNSW-NB15. Multiclass accuracy: up to 99.83% (NSL-KDD) and 99.99% (UNSW-NB15) with sampling. Their contribution: Demonstrated the effectiveness of the CNN-BiLSTM architecture with sampling for reliable anomaly detection in imbalanced datasets. However, it requires a large volume of normal network data to adequately learn the expected behavior of LSTM models.

Quamar *et al.* [3] propose a hybrid deep learning model for intrusion detection in IoT networks, combining a deep autoencoder with LSTM layers trained on the NSL-KDD dataset. The model aims to exploit the strengths of both components: the autoencoder's ability to learn a robust representation of normal data, coupled with the temporal ability of LSTMs to capture sequential dynamics. Experiments show superior performance in accuracy, recall, or F1-score, compared to conventional architectures, while maintaining a reasonable computational cost. However, despite its promising performance, the hybrid model proposed has some limitations. First, training deep networks combining autoencoders and LSTMs is computationally expensive, which may limit their deployment on low-capacity IoT devices. Second, reliance on datasets such as NSL-KDD, which do not fully reflect the diversity of current IoT scenarios, may result in limited generalization in real-world conditions. Finally, the model could be susceptible to adversarial attacks due to the lack of explicit defense mechanisms or robustness to noisy data.

Al-Qatf *et al.* [4] propose a network intrusion detection model (STL-IDS) based on the self-taught learning (STL) framework. This model starts with a sparse autoencoder used for feature learning and dimension reduction in an unsupervised mode. The resulting new representations are then fed to an SVM classifier, which

improves detection accuracy and reduces training and testing times. The system's performance is evaluated in binary and multi-class classifications, compared to conventional methods such as J48, naive Bayes, Random Forest, or simple SVM. The results show that the STL-IDS approach outperforms most existing methods, both in accuracy and speed, for network intrusion detection. However, the approach is not designed for real-time environments, as the preprocessing process and the classification phase (with SVM) can become costly at large scale or in time-constrained contexts.

Jang-Jaccard *et al.* [5] used an autoencoder (AE)-based model and a novel outlier removal method to avoid bias due to data imbalance for each input sample. Their approach improves reconstruction quality by filtering out the most influential outliers, thereby increasing the model's robustness to variations in network traffic. When evaluated on the NSL-KDD dataset, their model achieved an accuracy of 90.61% and an F1-score of 92.26%, outperforming several existing anomaly detection methods.

T. Su, H. Sun, J. Zhu *et al.* [6] proposed the BAT method, which consists of BLSTM (Bidirectional-LSTM) and attention, achieving an accuracy of 84.25% on the NSL-KDD test dataset.

Wang, H., and Li, W. [7] developed a hybrid neural network framework (DosTC) that combines efficient and scalable transformers with a CNN (Convolutional Neural Network) to detect DDoS (Distributed Denial of Service) attacks on SDN, which was evaluated on the CICDDoS219 dataset.

Elsayer *et al.* [8] proposed DDoSNET to detect distributed denial of service (DDoS) attacks in Software Defined Networks (SDN). They leveraged deep learning (DL) to develop their method, which is based on a recurrent neural network (RNN) with an autoencoder. They evaluated their model on the CICDDoS2019 dataset.

After establishing this framework, we move on to the methodological description of our approach in the next section.

3. Methodology

In this section, we discuss the methodology used to conduct our project. We begin by describing the dataset used, highlighting its specific characteristics and the data preprocessing applied. We then present the model architecture, algorithms, and libraries we selected to implement our solution. Finally, we will discuss the evaluation metrics we plan to use to measure the performance and effectiveness of our solution. These metrics will allow us to quantify our model's ability to detect intrusions and resist attacks.

3.1. Detection Workflow in AE-LSTM

Figure 1 describes the steps in the process for detecting attacks in IoT networks. It begins by using data from the IoT-23 and NSL-KDD datasets, followed by careful data cleaning, and then transforming them into time sequences. These se-

quences are then fed into our model, which consists of an autoencoder (AE) and LSTM. After training, performance metrics are used to evaluate the results. The reconstruction error is used to detect anomalies, and figures (graphs) are generated to analyze and interpret the results.

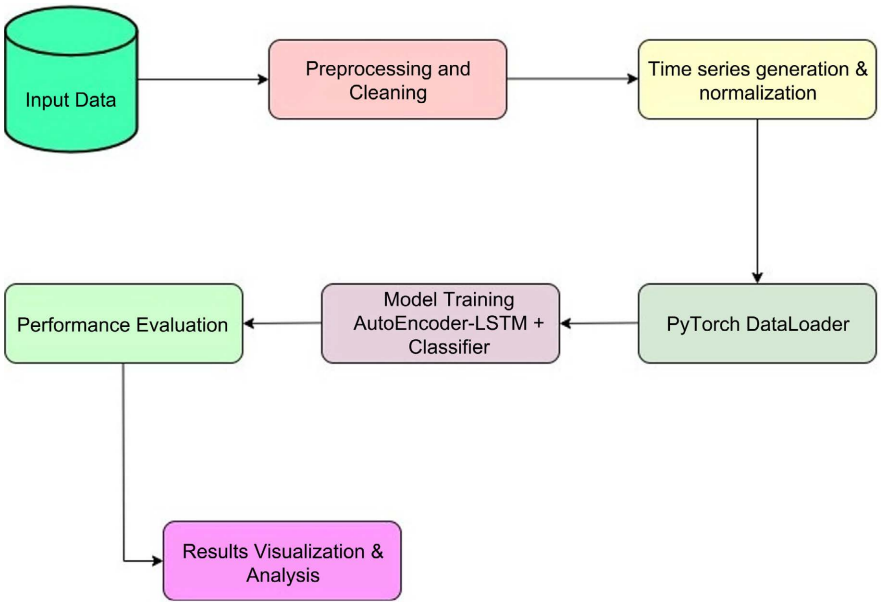

Figure 1. AE-LSTM model workflow.

3.2. Model Selection and Justification

AE-LSTM combines both the power of unsupervised reconstruction by detecting anomalies through reconstruction errors and temporal modeling with LSTM, which is a double advantage over zero-day attacks and specific attacks.

The autoencoder (AE) learns a compact latent representation of normal traffic by reducing dimensions and filtering noise. This latent compression allows anomalies to be detected effectively by comparing the input with its reconstruction. LSTMs are capable of memorizing long-term dependencies and managing the gradient effect, making them suitable for detecting anomalies in IoT flows.

This model is suitable for smart IoT environments because it reduces the complexity of the massive and heterogeneous data generated, detects temporal behaviors, and improves the detection of dynamic attacks originating from network traffic. This makes the AE-LSTM model more robust than autoencoder or LSTM models used alone.

3.3. Characteristics of the AE-LSTM Model

1) Encoder and Decoder:

The encoder is a 3-layer LSTM with a hidden layer size of 64 units. It transforms each input sequence into a compact latent vector representing the essential characteristics of network traffic. Only the last hidden layer (h_n[−1]) is used as the final representation. The latent vector x_enc = h_[−1] corresponds to the last hid-

den layer of the encoder LSTM. It is replicated at each time step to form the decoder input. This vector encodes the global summary of the sequence.

The Decoder (*LSTM*): It attempts to reconstruct the original sequence from the summary obtained. If the reconstruction is poor (using a measure called MSE), this may indicate an anomaly. The decoder is also a 3-layer LSTM, but inverted: it takes the latent vector (replicated) as input and attempts to reconstruct the original sequence. It measures the model's ability to memorize and reconstruct traffic behavior. The reconstruction loss is calculated using the MSE-Loss (Mean Squared Error) function. It measures the difference between the original sequence and the sequence reconstructed by the decoder. A faithful reconstruction means that the sequence is "normal," while a high difference may indicate an anomaly or an attack.

2) The optimizer and tuning: The model is trained using the Adam optimizer with the following parameters:

- Initial learning rate: 0.001, commonly recommended value for Adam.
- Regularization parameter (weight decay): 0.0001 to limit overfitting. which can cause poor generalization and erroneous anomaly detection.
- Total loss function: MSE + CrossEntropy, implicitly weighted equally.

These hyperparameters have been empirically validated based on performance curves on the validation set (stable loss and accuracy).

3) Training hyperparameter:

- Number of epochs: 100.
- Batch size: 128.
- Shuffle enabled in DataLoader to ensure mini-batch diversity.
- Simple cross-validation performed at each epoch (split 80% train/20% val).

4) Performance evaluation: The results are analyzed using standard classification metrics: Accuracy, Precision, Recall, F1-score, ROC curve for binary tasks, Confusion matrix for multi-class tasks, Loss and accuracy evolution curves during training.

5) Experimentation scripts: The entire process is organized into modular Python scripts:

a) preprocess.py: processing, transformation, and export of sequences.

b) LSTMAE.py: definition of the AutoEncoder architecture (encoder/decoder).

c) train.py: training loop, saving models, loss tracking.

d) evaluate.py: calculation of metrics on test data.

e) visualize.py: generation of figures (curves, matrices, etc.).

3.4. Description of Datasets

The datasets are used to evaluate dimensional reduction and anomaly detection using the autoencoder, capture of temporal dependencies using LSTM. Hybrid capability allowing both unsupervised detection and supervised classification model, in order to determine whether it can be used to accurately detect attacks or not. They may be the result of an IoT simulation environment or a real IoT environ-

ment. The quality of the dataset is very important and affects the performance of any intrusion management system in IoT network traffic.

3.5. Dataset Description

Datasets are used to evaluate the model to determine whether it can be used to accurately detect attacks. They can be the results obtained from a simulated IoT environment or a real IoT environment. The quality of the dataset is very important and affects the outcome of any intrusion detection system in IoT network traffic.

1) Dataset: NSL-KDD [9]

The NSL-KDD dataset is a widely used benchmark for evaluating intrusion detection systems. It contains 41 attributes describing network traffic, including 38 continuous attributes (duration, size, number of packets, etc.) and 3 discrete attributes (protocol, service, connection type). The data is labeled according to different attack types: 22 in the training set and 38 in the test set, which allows us to test the model's ability to detect novel attacks. This is an improvement of the KDD Cup '99 dataset created by DARPA in 1999 from network traffic records from the 1998 dataset [10], conducted by the MIT Lincoln Laboratory.

2) Dataset: IoT-23 [11]

Dataset: IoT-23 is a new dataset of network traffic from Internet of Things (IoT) devices. It features 20 captures of malware running on IoT devices and 3 captures of benign IoT device traffic. It was first published in January 2020, with captures ranging from 2018 to 2019. This IoT network traffic was captured in the Stratosphere Lab, AIC Group, FEL, CTU University, Czech Republic. Its goal is to provide a large dataset of realworld, labeled IoT malware infections and harmless IoT traffic for researchers to develop machine learning algorithms. This dataset and its research are funded by Avast Software, Prague.

3.6. Data Preprocessing

This is a set of steps that results in a final dataset that is clean, consistent, and usable by deep learning models.

3.6.1. For Our AE-LSTM Model Applied to the NSL-KDD Dataset

First, the raw CSV file is loaded. Then, some text columns, such as label, protocol_type, and flag are cleaned to make them consistent (removing spaces, converting to lowercase, etc.). Depending on the chosen training method, the labels are transformed in two ways: Since machine learning models can only process numeric values, it is necessary to convert categorical variables into usable numerical representations.

To do this, we use one-hot encoding to transform categorical variables such as protocols and services contained in the datasets into numerical vectors without introducing ordinal bias.

The benefit of this one-hot encoding is important because it ensures that each category is treated equally, without assuming an ordering relationship between

them, which is essential to preserve the neutrality of the models during training. The categorical variables contained in the NSL-KDD dataset are protocol type: tcp, udp, icmp, services: http, ftp, telnet, flags: SF, S0, REJ, and in the IoT-23 dataset we have proto: tcp, udp, icmp, conn state: SF, S0, REJ, and services: sometimes present depending on the parsing.

Next, we will need to remove rare categories and similar groups (duplicates). In addition to on-hot encoding, we use embedding to represent categorical variables in a continuous and compact manner and capture semantic similarities between categories, which improves generalization and reduces computational complexity. Depending on the chosen training method, the labels are transformed in two ways:

- In binary mode, the data is labeled into two classes: 0 for normal traffic and 1 for any form of attack.
- In multiclass mode, attacks are grouped into four broad categories (DoS, Probe, R2L, U2R) based on the attack name. Very rare or ambiguous attacks can be excluded to avoid imbalances or ambiguity in training.

After this clustering phase, the text columns used as clustering keys are removed to avoid any redundancy in the data. Categorical variables such as protocol_type, service, and flag are then encoded using OneHot Encoding, transforming each possible value into a distinct binary vector. All columns are then converted to numeric format.

Finally, a rebalancing step is applied to address the imbalance between classes. This ensures better model performance across all classes, whether majority or minority. The resulting final data frame is ready to be used as the optimal input for the AE-LSTM architecture, which will benefit from normalized, balanced, and semantically clear data.

This process ensures that our AE-LSTM model can fully exploit the temporal and contextual patterns of IoT network traffic, while benefiting from a consistent and usable representation of the data.

3.6.2. Preprocessing Algorithm Applied to the NSL-KDD Dataset

Figure 2 represents the preprocessing process for the NSL-KDD dataset.

3.6.3. For Our IoT-23 AE-LSTM Model

Just as for the NSL-KDD dataset, the preprocessing process results in a final dataset that is clean, consistent, and usable by deep learning models. It consists of removing unlabeled rows, deleting rows containing missing values in critical fields (timestamps, ports, packets), and removing duplicates. Then, the columns are converted into usable numeric values. A correlation matrix analysis identified highly redundant columns (correlation greater than 0.95), which were removed to reduce dimensionality and avoid multicollinearity. Finally, a correlation heatmap was generated to visualize linear relationships between variables. Once the data is ready, we use our AE-LSTM model to select the most relevant features, which will then be optimized with tools such as Adam, dropout, and early stopping.

```
Input:
    LOAD CSV "COOTRraint<.txt"
    1 - binarize : Boolean
Output: preprocessed and balanced DataFrame

1. LOAD CSV path INTO df
2. DELETE "difficulty" column FROM df
3. Clean text columns in df ["label", "protocol_type", "flag"]
4. IF binarize THEN
    IF binarize = TRUE
        FOR each row in df DO
            IF ("normal" is normal)
                df['label'] = 0
        END FOR
    ELSE
        FOR each row in df
            Categorize attacks into classes: 'DoS', 'Probe', 'R2L', or
            label to lowercase
            df['label'] != 0
            df['smurf'] if 'neptune' = 1
            df['satan'] if 'nmap' = 2
            df['ftp_write'] if 'phf' = 3
            df['clash'] != 3
            IF CASE
                Convert to 'DoS', 'Probe', 'R2L' or 'U2R'
                IF class == 1
                    df.remove this column 'class'
            END IF
5. DROP text labels
6. APPLY One-Hot Encoding on df ['protocol_type', 'service', 'flag']
7. CONVERT features TO numeric IN df
8. APPLY class balancing IN df
9. RETURN df
```

Figure 2. Preprocessing algorithm applied to the NSL-KDD dataset.

3.6.4. Algorithm for the Preprocessing Process on the IoT-23 Dataset

Figure 3 represents the preprocessing process of the IoT-23 dataset.

```
Input :
    - scenario : string ("mirai", "infiltration", "gagfyt" or other)
    - df : dataset containing a 'detailed-label' column
Output :
    - df : dataset with an updated 'label' column

1: IF scenario = "mirai" THEN
2:     FOR each row in df DO
3:         IF "bruteforce" is in detailed-label (lowercase) THEN
4:             df['label'] ← 1
5:         ELSE IF "ddos" is in detailed-label (lowercase) THEN
6:             df['label'] ← 3
7:         ELSE
8:             df['label'] ← 1
9:         END IF
10:     END FOR
11: ELSE IF scenario = "infiltration" THEN
12:     FOR each row in df DO
13:         df['label'] ← 2
14:     END FOR
15: ELSE IF scenario = "gagfyt" THEN
16:     FOR each row in df DO
17:         IF "ddos" is in detailed-label (lowercase) THEN
18:             df['label'] ← 4
19:         ELSE IF "c&c" is in detailed-label (lowercase) THEN
20:             df['label'] ← 1
21:         ELSE
22:             df['label'] ← 0
23:         END IF
24:     END FOR
25: ELSE
26:     FOR each row in df DO
27:         df['label'] ← 0
28:     END FOR
29: END IF
30: RETURN df
```

Figure 3. Preprocessing process algorithm applied to the IoT-23 dataset.

3.7. LSTM-AE Model

The AE-LST model is a hybrid model combining a sequential autoencoder based on LSTM (Long Short-Term Memory) networks for network flow reconstruction for intrusion detection. It allows robust latent representations to be extracted while distinguishing normal behavior from attacks, using a combined loss function.

3.7.1. Autoencoder (AE)

It is a deep neural network structure that trains to reduce the amount of data needed to represent input data. They are commonly used in machine learning to perform data compression, anomaly detection, or data reconstruction tasks [12]. Composed of an encoder at the input and a decoder and hidden layers at the output.

Figure 4 shows the encoding and decoding process of the autoencoder.

```
Input: Data y, latent_dim d, learning_rate η, epochs E
Output: Encoder, Decoder

Initialize enc_W, dec_W, enc_b, dec_b randomly

for epoch = 1 to E do
    for each sample y in Y do
        # Encode
        h = enc_f(enc_W * y + enc_b)
        z = enc_f(W_z * h + b_z)

        # Decode
        h' = dec_f(dec_W * z + dec_b)
       y' = dec_f(out_W * h' + out_b)

        # Loss
        L = MSE(y, y')

        # Backpropagation
        Update weights enc_W, dec_W, ... with gradient
descent
    end for
end for

return Encoder, Decoder
```

Figure 4. Autoencoder algorithm.

3.7.2. Long Short-Term Memory (LSTM)

This is an essential component of the AE-LSTM model. It captures the temporal dependencies of the input sequences via three (3) LSTM layers in the encoder and three (3) in the decoder. The last hidden state (h_n[−1]) is extracted as a latent vector, representing a summary of the sequence. This vector is repeated for each time step and fed to the decoder, which attempts to reconstruct the original sequence. The objective is to minimize the reconstruction error (via MSELoss), allowing for the detection of anomalies when the error exceeds a predefined threshold.

In our AE-LSTM model, LSTM captures the temporal structure of smart IoT environments in terms of successive packets. The LSTM encoder extracts compact

latent representations and the LSTM decoder reconstructs the original sequences. This improves the robustness of the autoencoder as it captures long-term dependencies and distinguishes normal sequences from attack sequences.

The architecture uses an adjustable hidden layer size (64 units) and can include a sigmoid activation for the output. The last hidden layer (h_n[−1]) is used as the final representation. The latent vector x_enc = h_[−1] corresponds to the last hidden layer of the encoder LSTM. It is replicated at each time step to form the input to the decoder. This vector encodes the overall summary of the sequence.

Figure 5 shows the LSTM encoding and decoding process.

```
Begin

# === Encoder Initialization ===
Define a 3-layer LSTM encoder with:
    - input_size = dimensionality of input features
    - hidden_size = 64
    - num_layers = 3
    - dropout = 0.3
    - batch_first = True

# === Encoding Phase ===
Apply encoder to the input sequence x:
    enc_out, (h_n, c_n) = encoder(x)

Extract the latent vector from the last hidden state of the final (3rd) encoder layer:
    x_enc = h_n[-1]     # Shape: (batch_size, hidden_size)

# === Latent Vector Replication ===
Repeat the latent vector across the sequence length:
    z = repeat(x_enc, seq_len times)
    # Shape: (batch_size, seq_len, hidden_size)

# === Decoder Initialization ===
Define a 3-layer LSTM decoder with:
    - input_size = hidden_size  (same as encoder output)
    - hidden_size = same value as encoder
    - num_layers = 3
    - dropout = 0.3
    - batch_first = True

# === Decoding Phase ===
Apply decoder to the replicated latent vector:
    dec_out, _ = decoder(z)

# === Output ===
The reconstructed sequence is:
    dec_out   # Shape: (batch_size, seq_len, hidden_size)

End
```

Figure 5. LSTM encoder algorithm.

3.7.3. Performance Metrics

Performance metrics are used to evaluate our AE-LSTM model in intrusion management.

1) Reconstruction Error: Measures the discrepancy between the input data and its reconstruction, used to detect anomalies.

$$\text{ReconstructionError} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \hat{x}_i\right)^2 \tag{1}$$

2) Accuracy (Correct Classification Rate): This indicates the overall proportion of correct predictions across the entire dataset.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + TP + TN} \quad (2)$$

3) Precision: It measures the proportion of true positive predictions among all positive predictions. Formula (per class):

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

4) Recall: It measures the model's ability to detect all true attacks.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

5) F1-Score: It balances precision and recall.

$$\text{F1Score} = \frac{2 \times \text{precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

4. Results and Discussion

In this section, we present and discuss the experimental results obtained from our AE-LSTM model, trained and tested on the NSL-KDD and IoT-23 datasets.

4.1. Model Results on the NSL-KDD Dataset

In the article by Mahmoud *et al.* [1], the NSL-KDD dataset is structured around five main classes: Normal, DoS, Probe, R2L and U2R. The authors report a strong dominance of the Normal (49.39%) and DoS (36%) classes, which reflects a significant imbalance in the sample distribution. In our own study, conducted on the same dataset, we obtained a more balanced distribution between classes, with a moderate predominance of Normal (50%), followed by DoS (22%), Probe (13%), R2L (12%), and U2R (3%). This distribution allows for better representation of rare attacks and contributes to improving the robustness of detection models, particularly in multiclass scenarios.

4.1.1. Distribution between Normal and Attack Traffic on NSL-KDD Dataset

Figure 6 shows the balanced distribution of attack categories and normal data in the NSL-KDD dataset after processing. It helps visualize how the data is distributed across classes (Normal, DoS, Probe, R2L, U2R) for the AE-LSTM model. Since this model relies on reconstruction error to detect anomalies, a balanced distribution is crucial to ensure it learns to reconstruct normal data and identify deviations caused by attacks.

4.1.2. Binary Classification Loss vs. Epoch Plot on NSL-KDD Dataset

Figure 7 plots the average loss (MSE) per epoch during training. The model calculates the difference between the reconstructed inputs and outputs for each batch, and the loss is recorded for each epoch. It corresponds to the model's error during prediction:

- A progressive decrease in loss (in red) indicates that the model is improving in

reconstructing normal sequences.

- If the loss stagnates or increases after an initial decrease, this may indicate that the model is having difficulty generalizing or is encountering complex data (e.g., rare attacks).
- A low and stable loss demonstrates a good ability to model normal data, essential for detecting anomalies.

The "Train Loss" curve indicates the learning progress, while "Val Loss" reflects the generalization ability. A continuous decrease followed by a stagnation or increase in the valid loss may justify early stopping.

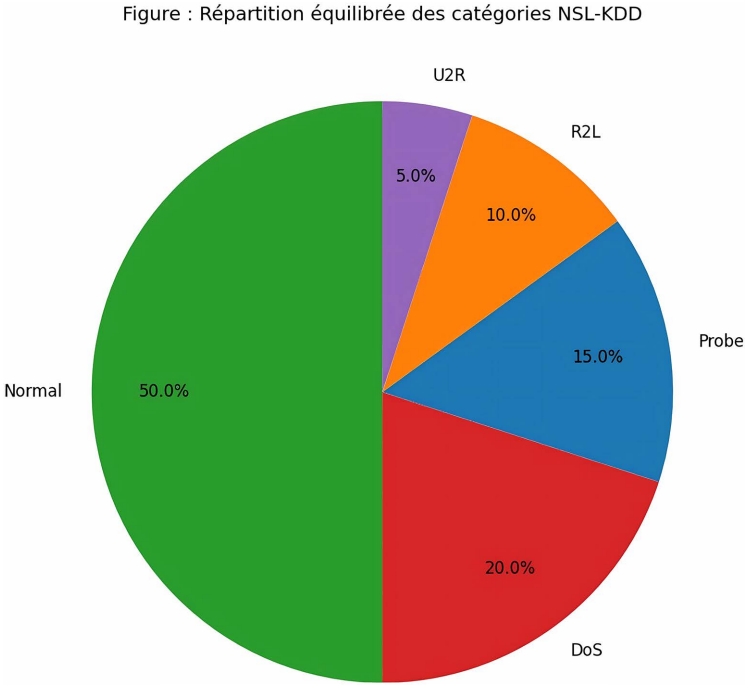

Figure 6. Attack distribution on NSL-KDD dataset.

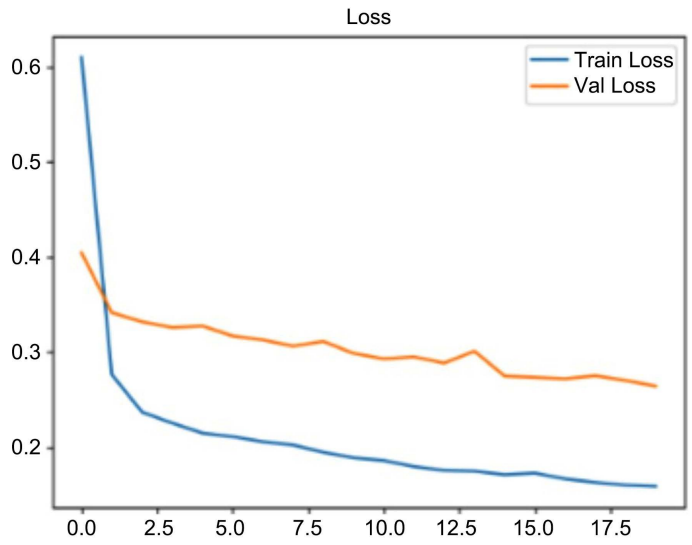

Figure 7. Loss vs. epoch graph on NSL-KDD dataset.

4.1.3. Binary Classification for Accuracy vs. Epoch Curve on NSL-KDD Dataset

Figure 8 illustrates the evolution of the model's accuracy during training and validation. Accuracy measures the model's ability to distinguish normal data from anomalies using the reconstruction error as a metric.

- A high accuracy (close to 100%) suggests that the chosen threshold works well in separating normal data from attacks.
- If the accuracy is low, it may indicate that the threshold is poorly adjusted or that the reconstruction errors do not differentiate the classes well.

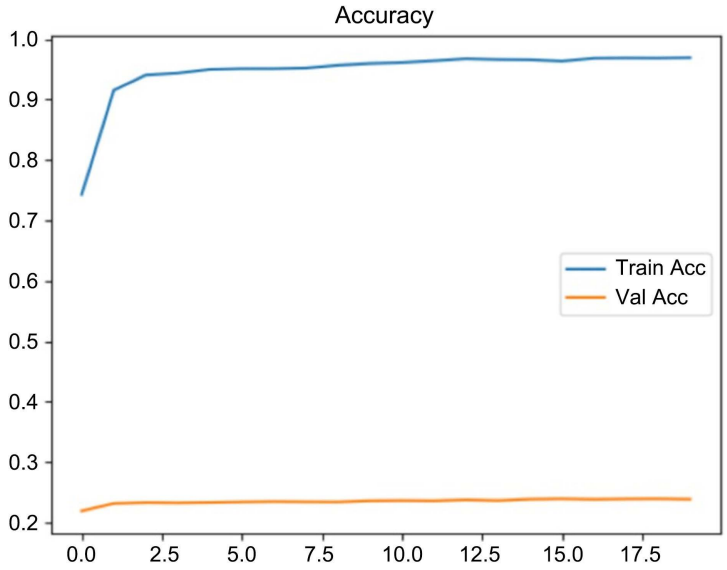

Figure 8. Accuracy vs. epoch graph on NSL-KDD dataset.

4.1.4. Binary Classification Confusion Matrix on NSL-KDD Dataset

Figure 9 visualizes the model's performance by showing the number of correct and incorrect predictions (true/false positives/negatives) for detecting anomalies. It constructs a binary confusion matrix (Normal vs. Attack) using the reconstruction errors and a threshold. The true labels are simplified (Normal = 0, Attack = 1), and predictions are based on whether the threshold is exceeded.

- Each row corresponds to a real class.
- Each column represents a predicted class.
- The cells display the number and percentage of predictions for each combination.
- A matrix with a high concentration on the diagonal suggests good anomaly detection, while high off-diagonal values indicate confusion.

This matrix allows us to identify:

Well-distinguished classes (strong diagonal), the frequent confusion between certain classes (off-diagonal values), and possible imbalances or weaknesses of the model on rare classes.

It is essential, particularly in multiclass classification, to see how the model handles categories like U2R or R2L, which are often underrepresented.

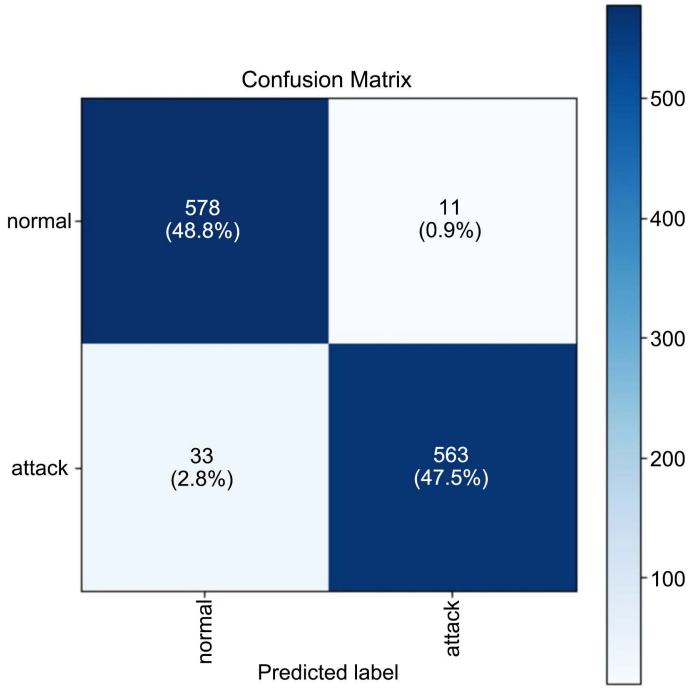

Figure 9. Binary confusion matrix on NSL-KDD dataset.

4.1.5. Multi-Class Classification Loss Curve on NSL-KDD Dataset

Figure 10 shows how the AELSTM model learns to reconstruct data and classify categories (Normal, DoS, Probe, R2L, U2R) over time. This is a key indicator for assessing whether training is effective and whether the model is avoiding overfitting.

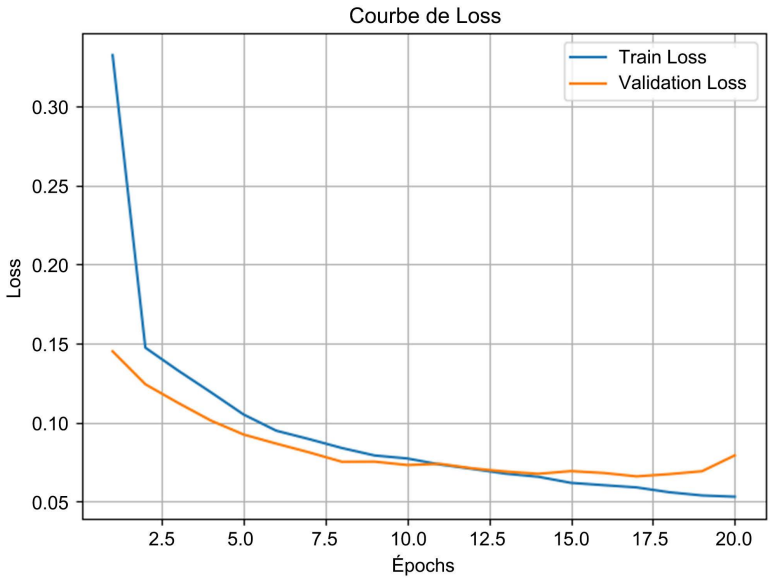

Figure 10. Loss vs. epoch graph on NSL-KDD dataset.

If the "Train Loss" (in blue) and "Validation Loss" (in orange) curves gradually decrease, this means that the model is improving its reconstruction and classifi-

cation. If the "Train Loss" decreases but the "Validation Loss" increases or remains stagnant, this may indicate overfitting (the model memorizes the training data but does not generalize well). A loss stable at a low value (close to 0) shows that the model has converged well and performs well on the five NSL-KDD classes.

A loss curve that decreases well on both sides is a clear indicator that the model is making progress in learning reconstruction and classification without overfitting. This is a good sign of generalization.

4.1.6. Multi-Class Classification Accuracy Curve on NSL-KDD Dataset

Figure 11 measures the model's ability to correctly predict the five classes (Normal, DoS, Probe, R2L, U2R) in the training and validation data.

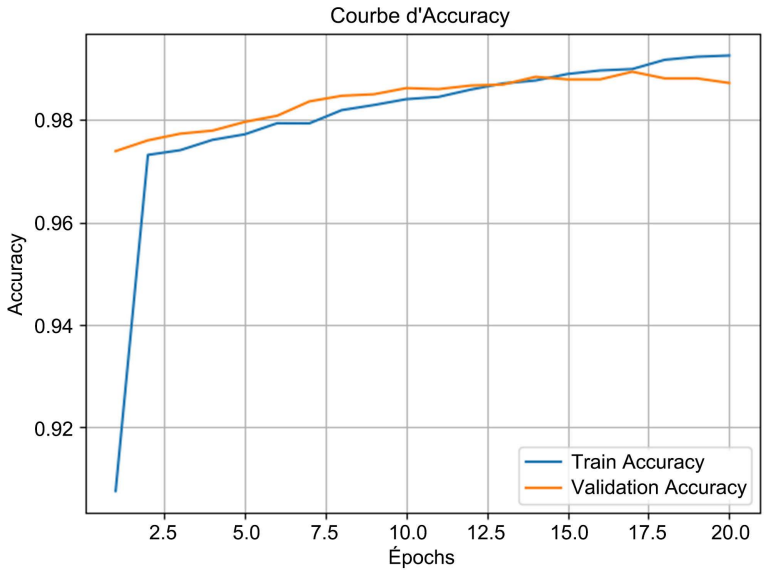

Figure 11. Precision versus epoch graph on multiclassification on NSL-KDD dataset.

- An upward-sloping accuracy curve indicates that the model is learning to classify better and better.
- An increase in "Train Accuracy" (in blue) and "Validation Accuracy" (in orange) indicates that the model is becoming more accurate over time.
- If the accuracy on the training set is very high but the accuracy on the validation set is low or stagnant, this means that the model is not generalizing.

An increasing and stable accuracy, especially during validation, indicates that the AE-LSTM model is capable of successfully detecting and differentiating the different intrusion classes in the NSL-KDD dataset. This is a good performance indicator for a multiclass IDS.

4.1.7. Multi-Class Classification Confusion Matrix on NSL-KDD Dataset

Figure 12 compares the actual values (in rows) with the values predicted by the model (in columns). Each cell indicates the number of samples correctly or incorrectly classified for each attack category.

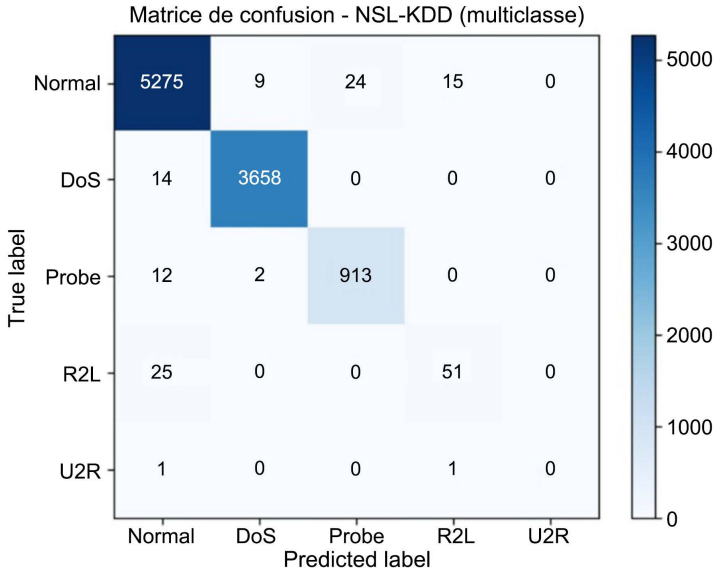

Figure 12. Multi-classification confusion matrix on NSL-KDD dataset.

The confusion matrix confirms that the model correctly classifies the majority of attacks, despite difficulty with underrepresented classes. This reflects excellent overall performance for a multiclass intrusion detection system on the NSL-KDD dataset.

4.1.8. ROC Curve (Receiver Operating Characteristic) on NSL-KDD Dataset

Figure 13 visualizes the distribution of reconstruction errors for the different NSL-KDD categories (Normal, DoS, Probe, R2L, U2R). The figure helps assess whether the reconstruction error can effectively separate normal data from anomalies, which is the basis for detection in this type of model.

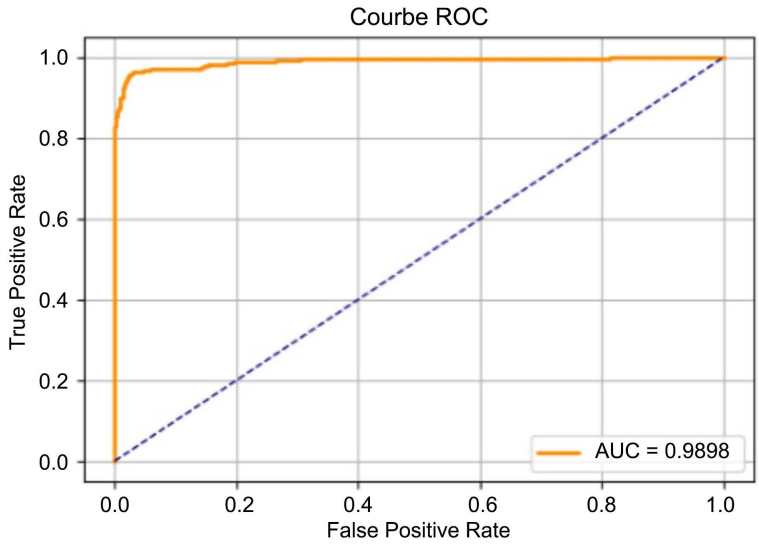

Figure 13. ROC curve on NSL-KDD dataset.

This curve plots the True Positive Rate (TPR) as a function of the False Positive

Rate (FPR) for different decision thresholds.

The model, with six LSTM layers (three encoders + three decoders), is trained on the normal data of the NSL-KDD dataset to learn how to reconstruct the sequences.

After training, the reconstruction error (usually the mean squared difference, MSE) is calculated for each sequence in the validation set, which includes both normal data and attacks (DoS, Probe, R2L, U2R). The figure plots these errors as a box plot or a distribution (histogram) by category (Normal vs. Attacks), with a line indicating a potential threshold for detecting anomalies.

- If the boxplot or histogram shows that the errors for "Normal" are significantly lower (e.g., median around 0.01) than for attacks (e.g., median around 0.1 for DoS), this indicates that the model reconstructs normal data well and can detect anomalies.
- A strong overlap between the errors for "Normal" and those for attacks as Probe or R2L suggests that the model is having difficulty differentiating certain anomalies, which might require adjustment (more epochs, better threshold).
- The chosen threshold of 0.05 should be placed where the errors for "Normal" are mostly below this threshold and those for attacks are mostly above this threshold. For example, if 95% of the errors for "Normal" are <0.05 and 80% of the errors for "DoS" are >0.05, this threshold would be effective.
- Extreme values (outliers) in attacks, such as U2R, may be rare, but they may indicate cases that are difficult to detect, reflecting the difficulty of modeling underrepresented classes.

We observe that the curve strongly approaches the upper left corner, which reflects excellent discrimination between the "normal" and "attack" classes.

The area under the curve (AUC) is equal to 0.9890, indicating near-perfect performance. An AUC of 1.0 represents perfect separation, while an AUC of 0.5 corresponds to a random model. Thus, with AUC = 0.9890, our model exhibits excellent robustness in binary classification, capable of detecting intrusions while limiting false alarms.

Table 1 above presents the overall performance of our AE-LSTM model applied to the NSL-KDD dataset. The results demonstrate excellent intrusion detection performance, with precision and recall above 99%, and a balanced F1 score of 99.25%, confirming the effectiveness of our approach on both normal and attack classes.

Table 1. Performance of the AE-LSTM model on the NSL-KDD dataset.

Metric	Value (%)
Accuracy	98.89
Precision	98.88
Recall	98.88
F1-score	98.88

4.2. Model Results on IoT-23

Contrary to the article by Mahmoud *et al.* [1], which is based on the NSL-KDD dataset structured around five classes (Normal, DoS, Probe, R2L, U2R) with a strong dominance of the "Normal" (49.39%) and "DoS" (36%) classes, our study uses the IoT-23 dataset, which is more representative of recent attacks targeting connected objects.

4.2.1. Distribution of Attack and Non-Attack Percentage on IoT-23

Figure 14 shows the distribution, notably a higher proportion of normal traffic (56.6%) and IoT-specific attacks such as Gagfyt (18.9%), Infiltration (21.3%), and Mirai brute-force (3.2%). This unbalanced distribution may justify the use of an autoencoder, which is mainly based on the reconstruction of normal traffic.

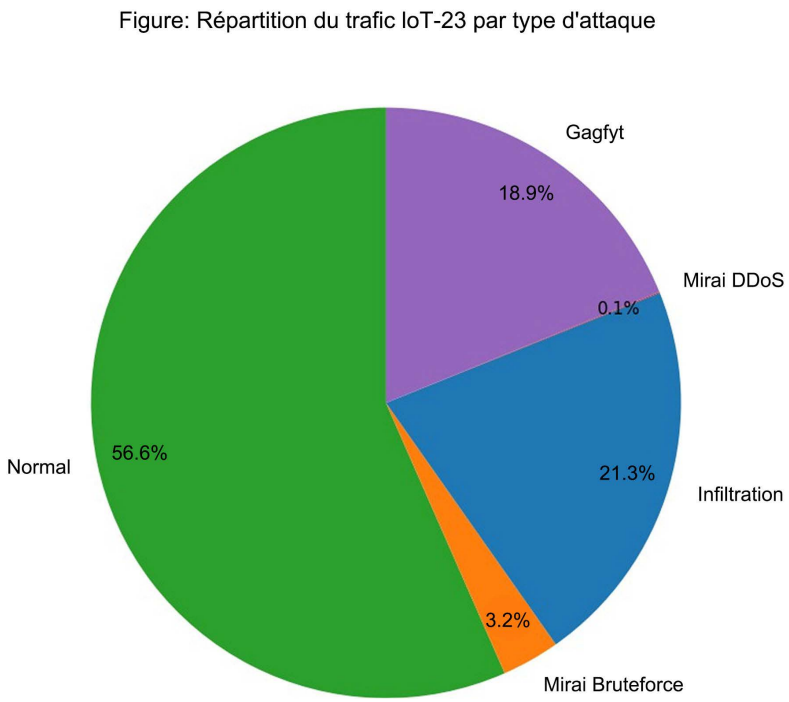

Figure 14. Attack distribution on IoT-23.

4.2.2. Binary Classification Loss versus Epoch Curve on IoT-23

Figure 15 shows that the gradual decrease reflects good reconstruction capacity and training without overfitting. The validation loss is lower than the training loss and decreases more steadily, which is a very good sign.

4.2.3. Binary Classification Accuracy Curve versus Epoch on IoT-23

Figure 16 shows the evolution of the accuracy of the AELSTM model over iterations. She reflects good reconstruction capacity and training without overfitting.

- The training accuracy starts around 96.5% at the very beginning, and quickly rises to 100% from the 2nd epoch.
- This shows that the model learns very quickly, which is typical for a well-regulated autoencoder.

We observe a very rapid convergence of the training accuracy, reaching almost 100% in the first epochs, while the validation accuracy remains stable at around 99.9%. This demonstrates the model's good generalization ability for anomaly detection in the IoT-23 dataset sequences.

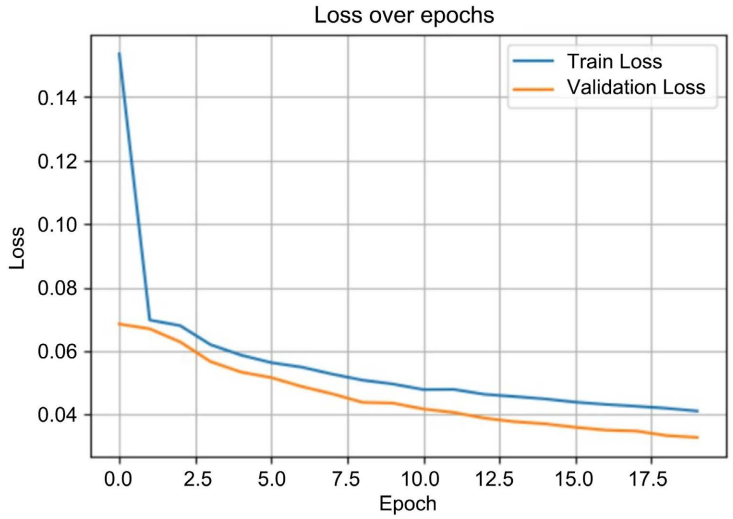

Figure 15. Loss versus epoch graphs for binary classification on IoT-23.

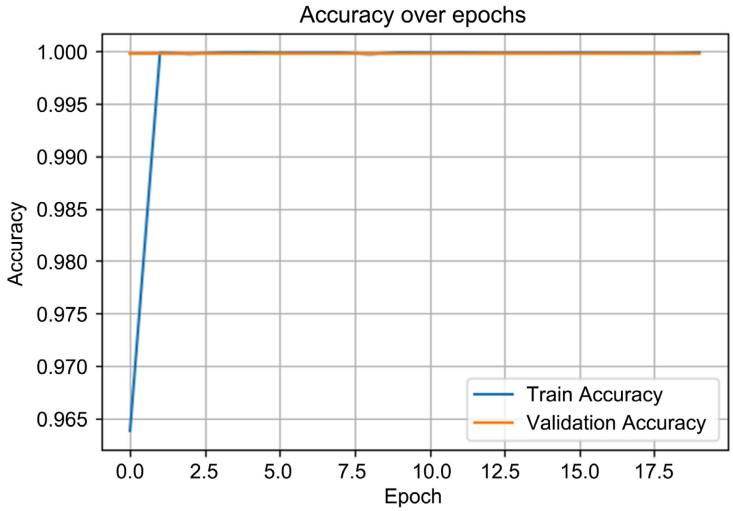

Figure 16. Accuracy graphs versus epoch for binary classification on IoT-23.

4.2.4. Binary Classification Confusion Matrix on IoT-23

Table 2 allows us to visualize the number of correct and incorrect predictions made by the model for the two classes: normal traffic (0) and malicious traffic (1). It is structured as follows:

- The vertical axis (True Label) indicates the actual classes in the test set.
- The horizontal axis (Predicted Label) corresponds to the classes predicted by the model.
- Each cell contains the number of samples classified in this way, followed by the percentage relative to the row total.

Table 2. Confusion matrix for binary classification on IoT-23.

Actual/Predicted class	Normal (0)	Attack (1)
Normal (0)	4954 (100%)	1 (0.0%)
Attack (1)	1 (0.0%)	5044 (100%)

Top left (4954, 100%): all actual samples in the normal class are correctly identified as normal, with one exception.

Bottom right (5044, 100%): almost all samples in the attack class are correctly identified, with one exception.

The values in red (1, 0.0%) indicate misclassifications (false positives or false negatives), which are extremely rare here.

This result reflects very high precision and recall for both classes, confirming the robustness of the model for intrusion detection in a binary context. The overall correct classification rate exceeds 99.98%, reflecting near-perfect system performance on this dataset.

The results presented in Table 2 show that our AE-LSTM model is reliable for binary classification. It correctly identifies almost all behaviours (Figure 17).

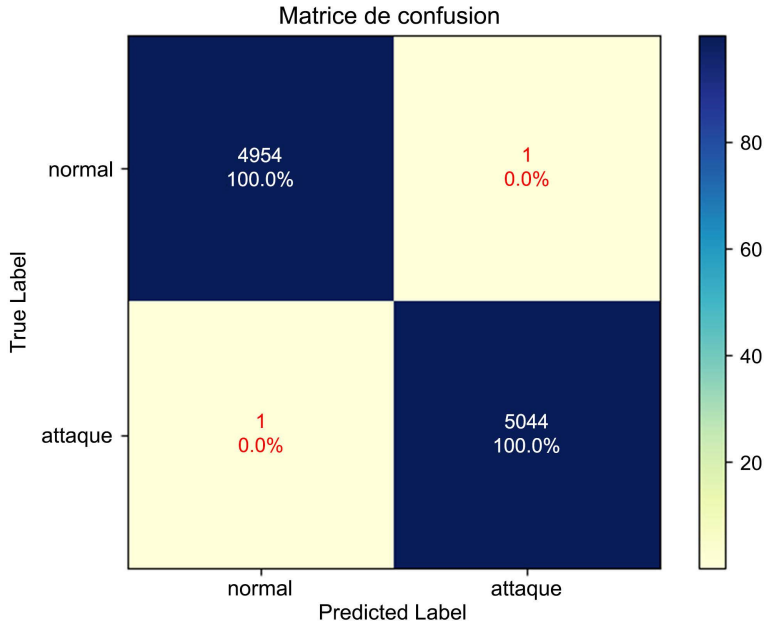

Figure 17. Confusion matrix for binary classification on IoT-23.

4.2.5. Multi-Class Classification Epoch Loss Curve on IoT-23

Figure 18 shows how the model learns to reduce its errors over time. A decrease in loss indicates that the model is improving at reconstructing the data (normal, mirai, infiltration, gagfyt). It is generated by the plot_losses function. It plots the evolution of loss over epochs for the training (Train Loss) and validation (Validation Loss) sets. The loss is calculated as the sum of the reconstruction loss (MSE) during training of the AE-LSTM model.

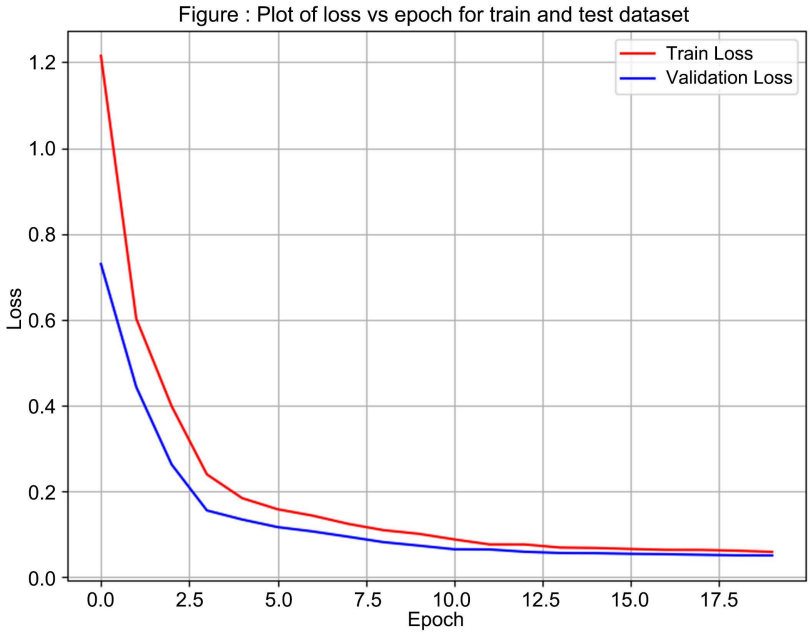

Figure 18. Loss versus epoch graphs for multiclass on IoT-23.

- An increase in "Train Accuracy" (in green) and "Validation Accuracy" (in orange) indicates that the model is becoming more accurate.
- If the "Validation Accuracy" stagnates or decreases while the "Train Accuracy" continues to increase, this may indicate overfitting.

4.2.6. Multi-Class Classification Accuracy Curve versus Epoch on IoT-23

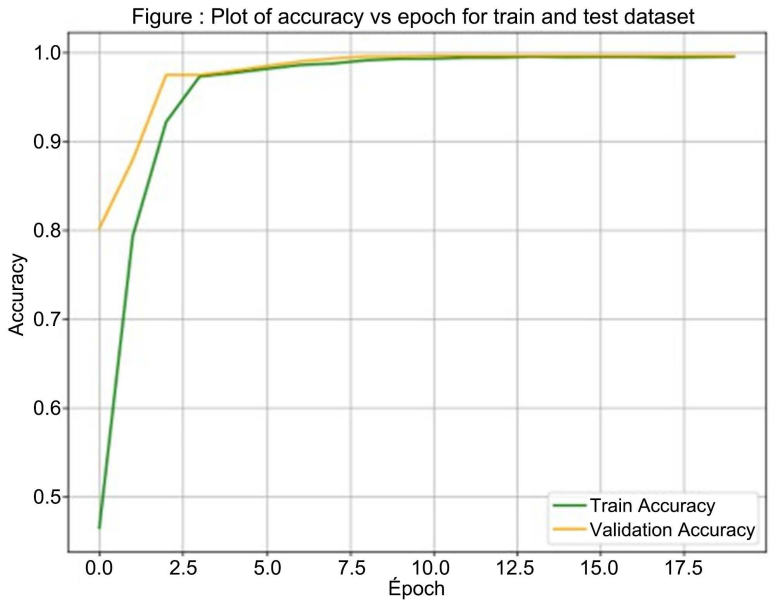

Figure 19. Accuracy graphs versus epoch for multiclassification on IoT-23.

Figure 19 measures the model's ability to correctly predict the classes (normal, mirai, infiltration, gagfyt) in the training and validation data. This is created by

the plot_accuracy function. It shows the evolution of accuracy over time for the training (Train Accuracy) and validation (Validation Accuracy) sets. Accuracy is calculated by comparing the model's predictions to the true labels.

The model becomes more accurate if Train Accuracy (in green) and Validation Accuracy (in orange) increase. However, if Validation Accuracy stagnates or decreases while Train Accuracy continues to grow, this may indicate overfitting.

4.2.7. Multi-Class Classification Confusion Matrix on IoT-23

Figure 20 shows the confusion matrix for multiclassification on the IoT-23 dataset. We obtain the following values: 2377 samples tested (sum of all cells) and 2377 – 9 = 2368 correctly classified for an overall accuracy of 99.6%. The AE-LSTM architecture is capable of classifying multi-class network traffic with high accuracy, including for attacks that are difficult to distinguish.

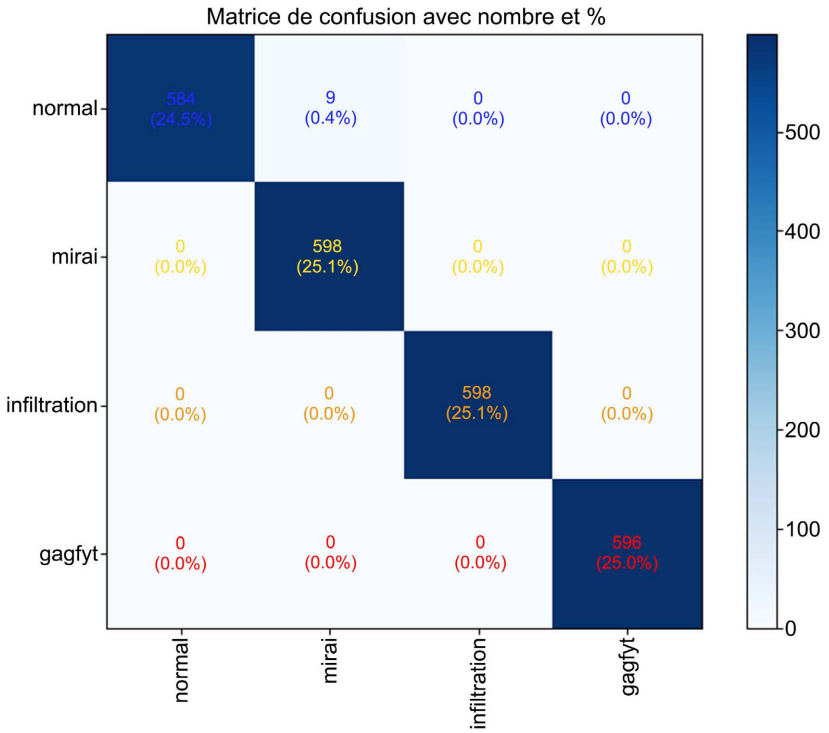

Figure 20. Confusion matrix for multi-classification on IoT-23.

4.2.8. The ROC Curve on IoT-23

Figure 21 represents the evolution of the True Positive Rate as a function of the False Positive Rate at different classification thresholds. The model achieves an area under the curve (AUC) of 0.9998, which means that it has a near-perfect ability to discriminate between normal traffic and attacks.

The AE-LSTM model is applied to the IoT-23 dataset, using a 6-layer LSTM architecture for time series encoding and decoding. This configuration allows for in-depth feature extraction from IoT network traffic data. Intrusion detection is based solely on reconstruction error, where significant discrepancies between input data and reconstructed data indicate anomalies.

Table 3 shows the remarkable performance achieved by our model with an accuracy of 99.02%, precision of 99.05%, recall of 99.07% and an F1 score of 99.09%. This performance demonstrates the robustness of the model in identifying malicious behavior in IoT environments.

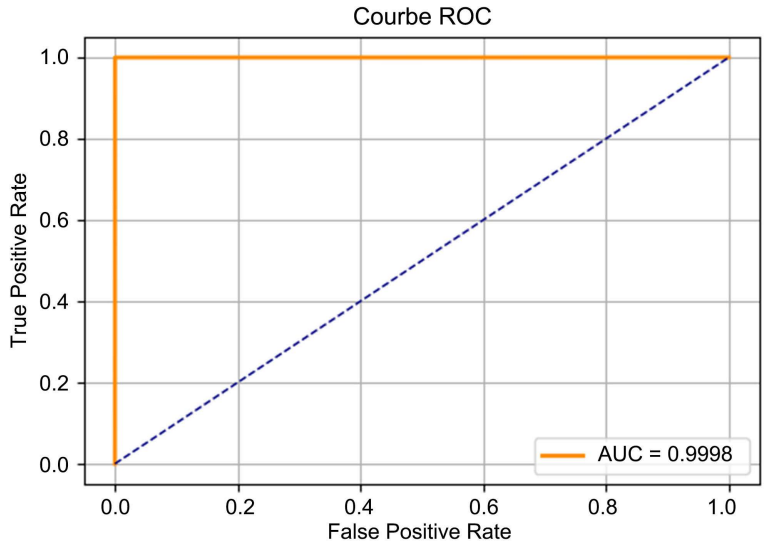

Figure 21. AE-LSTM ROC curves on IoT-23.

Table 3. Performance of the AE-LSTM model on the IoT-23 dataset.

Metric	Value (%)
Accuracy	99.02
Precision	99.05
Recall	99.07
F1-score	99.09

4.3. Comparison of the Two AE-LSTM Models

The AE-LSTM model applied to the NSL-KDD dataset demonstrates solid performance, with precision, recall, and F1-scores all around 98.88%, reflecting a good ability to detect anomalies via reconstruction error on a balanced set of five categories (Normal, DoS, Probe, R2L, U2R). In comparison, the AE-LSTM model with the IoT-23 dataset slightly outperforms these results with an accuracy of 99.02%, a precision of 99.05%, a recall of 99.07%, and an F1 score of 99.09%.

This slight superiority can be explained by the unbalanced distribution of IoT-23, which favors learning on normal data (56.6%), allowing for better reconstruction and detection of anomalies such as Infiltration or Gagfyt. However, NSL-KDD offers a more diverse coverage of attack types thanks to its balancing (50% Normal, 22% DoS, etc.), making it more robust against a variety of anomalies, although its performance is slightly lower. Both models demonstrate fast convergence and good generalization, but IoT-23 takes advantage of its data structure to achieve slightly higher metrics.

Table 4 shows that our model achieves near-perfect scores on IoT-23, including a binary F1-score of 99.98%, demonstrating a very low error rate. Multi-class performance is also better on IoT-23, with a remarkable ability to distinguish complex attacks such as Mirai or Gafgyt. On NSL-KDD, results remain excellent, but slightly lower, particularly for rare classes (U2R, R2L), illustrating the limitations of this dataset when faced with new types of threats.

Table 4. Binary performance comparison of AE-LSTM on NSL-KDD and IoT-23.

Metric	NSL-KDD	IoT-23
Accuracy	98.89 %	99.02 %
Precision	98.88 %	99.05 %
Rappel	98.88 %	99.07 %
F1-score	98.88 %	99.09 %

The **IoT-23** dataset proves to be more efficient than **NSL-KDD** for training intrusion detection models, particularly the *AE-LSTM* architecture, for several major reasons:

1) Regarding the timeliness and relevance of attacks, IoT-23 contains traces from realistic and recent cyberattack scenarios in IoT environments. However, NSL-KDD is derived from KDD'99 and includes obsolete or oversimplified attacks, which are less representative of current threats.

2) Regarding the diversity of attack scenarios, IoT-23 integrates a wide range of scenarios (botnet, DDoS, scanning, etc.), which enriches the model's training and improves its generalization and fine-grained recognition of abnormal behaviors.

Unlike NSL-KDD, which classifies attacks into general categories (DoS, Probe, R2L, U2R), IoT-23 offers more detailed labels to facilitate finer classification and improve model accuracy. Our AE-LSTM model exploits temporal relationships between packets or network flows. IoT-23 is better suited to this approach because it offers detailed time sequences, unlike NSL-KDD, whose data is essentially tabular and static.

NSL-KDD contains many redundant instances and a significant imbalance between classes. IoT-23 offers better structured, more balanced data, rigorously extracted from real captures (PCAP files transformed into enriched flows).

4.4. Discussion on the Limitations of the Model

The model was trained on two datasets, NSL-KDD (simulated environment) and IoT-23 (real environment), and the results obtained were analyzed and compared with those of other researchers in the literature. The results obtained are very encouraging: the model achieves an overall accuracy of over 99%, even in the presence of unbalanced or noisy classes. It has demonstrated a remarkable ability to distinguish between similar attacks and to generalize across a variety of cases.

These results obtained with the AE-LSTM model demonstrate promising performance for intrusion management in smart IoT environments, but several lim-

itations should be noted: our evaluation is performed on reference datasets (NSL-KDD and IoT-23), which, although useful for comparison, do not always reflect the variability and dynamics of real IoT production environments.

The model identifies anomalies via reconstruction error; however, its ability to generalize completely new attacks remains partially limited.

5. General Conclusions

The Internet of Things (IoT) is a rapidly growing technology that has transformed our lives, making them intelligent. In this article, we addressed the major challenges related to the security of Internet of Things (IoT) systems. To address the limitations of traditional intrusion detection approaches, which are often ineffective in the face of new or complex attacks, we designed a hybrid solution based on deep learning, combining two complementary modules: an LSTM autoencoder (AE-LSTM).

The model was trained on two datasets: NSL-KDD (simulated environment) and IoT-23 (real environment), and the results were analyzed and compared with those of other researchers in the literature. The results obtained are very encouraging: the model achieves an overall accuracy of over 99%, even in the presence of unbalanced or noisy classes. It demonstrated a remarkable ability to distinguish similar attacks and generalize across diverse cases.

The results obtained from training on two datasets: NSL-KDD (simulated environment) and IoT-23 (real environment) are very encouraging: the model achieves an overall accuracy of over 99%, even in the presence of unbalanced or noisy classes. It has demonstrated a remarkable ability to distinguish between similar attacks and to generalize across various cases.

As a perspective, our model can be integrated with attention modules (e.g., Transformer) to improve the capture of long-term dependencies; applied to production IoT environments, with real-world resource constraints.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Mahmoud, M., Kasem, M., Abdallah, A. and Kang, H.S. (2022) AE-LSTM: Autoencoder with LSTM-Based Intrusion Detection in IoT. 2022 *International Telecommunications Conference* (*ITC-Egypt*), Alexandria, 26-28 July 2022, 1-6. https://doi.org/10.1109/itc-egypt55520.2022.9855688

[2] Cosimo, A., *et al.* (2019) LSTM-Based Anomaly Detection in Network Traffic. *Computer Networks*, **159**, 147-166.

[3] Quamar, A., *et al.* (2021) Hybrid Deep Learning Model for Intrusion Detection in IoT Networks. *Proceedings of the* 2021 *International Conference on Cybersecurity*, 1-6.

[4] Al-Qatf, M., Lasheng, Y., Al-Habib, M. and Al-Sabahi, K. (2018) Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection. *IEEE Access*, **6**, 52843-52856. https://doi.org/10.1109/access.2018.2869577

[5] Jang-Jaccard, J., Xu, W., Singh, A., Wei, Y. and Sabrina, F. (2021) Improving Performance of Autoencoder-Based Network Anomaly Detection on NSLKDD Dataset. *IEEE Access*, **4**, 1-10.

[6] Su, T., Sun, H., Zhu, J., Wang, S. and Li, Y. (2020) Bat: méthodes d'apprentissage profond sur la détection d'intrusion réseau à l'aide de l'ensemble de données nslkdd. *IEEE Access*, **8**, 29575-29585. https://doi.org/10.1109/access.2020.2972627

[7] Wang, H. and Li, W. (2021) Ddostc: Un mécanisme hybride de détection d'attaque réseau basé sur un transformateur dans sdn. *Sensors*, **21**, Article 5047. https://doi.org/10.3390/s21155047

[8] Elsayed, M.S., Le-Khac, N.-A., Dev, S. and Jurcut, A.D. (2020) Ddosnet: Un modèle d'apprentissage profond pour la détection des attaques réseau. *Proceedings of the* 2020 *International Conference on Artificial Intelligence and Computer Engineering* (*ICAICE*), 54-60.

[9] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009) A Detailed Analysis of the KDD CUP 99 Data Set. 2009 *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, 8-10 July 2009, 1-6. https://doi.org/10.1109/cisda.2009.5356528

[10] Wenke, L. and Stolfo, S.J. (1999) KDD Cup 1999 Data. In Dataset Derived from DARPA 1998 IDS Evaluation by MIT Lincoln Laboratory.

[11] Garcia, S., Parmisano, A. and Erquiaga, M.J. (2020) IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic. Zenodo.

[12] Hinton, G.E. and Salakhutdinov, R.R. (2006) Reducing the Dimensionality of Data with Neural Networks. *Science*, **313**, 504-507. https://doi.org/10.1126/science.1127647