APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) p. 1037-1050
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Comparative Analysis of Anomaly-based Intrusion Detection Model in IoT Network

Lim Seng Chen[1], Kamaruddin Malik Mohamad[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: malik@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.056

Article Info

Received: 19 July 2025
Accepted: 15 June 2026
Available online: 30 June 2026

Keywords

Anomaly Detection, Intrusion Detection, Deep Learning, Internet-of-Things, Long Short-Term Memory, Autoencoder

Abstract

The selection of detection model for Intrusion Detection Sytem (IDS) is crucial to ensure network security, especially in IoT networks due to the constant internet connectivity and unique properties like heterogeneity. This research aims to study, to develop, to evaluate, and to compare two widely used deep learning anomaly detections models as intrusion detection models, which are: Long Short-Term Memory (LSTM) and autoencoder. The methodology framework includes pre-processing the IoT-23 dataset, developing, evaluating, and comparing the anomaly detection models in terms of accuracy, precision, recall, F1 score, and AUC-PR. The results indicate that LSTM performs better overall, but its strength is demonstrated by high precision of 0.986, meanwhile the autoencoder demonstrates its strength with high recall of 1.000. While the findings suggest LSTM is a more balanced approach and autoencoder is a more sensitive approach, future work should explore hybrid approach to combine their strengths to provide more robust detection.

1. Introduction

Intrusion detection is a useful security technique to detect malicious activities in a network traffic. Choosing a suitable detection model allows an intrusion detection system to work effectively in early threats detection. By detecting the threats early, data breaches can be prevented or taken appropriate actions. In IoT network, an effective intrusion detection is especially crucial as these networks are exposed to the risk of cybersecurity. This is due to the properties of the network such as the constant connection to the internet, device heterogeneity, and limited hardware resources [1]. These properties impose challenges in implementing security features and the properties of heterogeneity had complicates the selection of intrusion detection model [2].

The use of deep learning algorithms as intrusion detection models has been a research subject until today. There are many deep learning models that are used in anomaly detection. Two of the widely used deep learning models in anomaly detection include Long Short-Term Memory (LSTM) and autoencoder. Both of the models have the ability to learn pattern in normal data and identify anomalies once trained. Each detection model has their own strength and weaknesses. Therefore, this research aims to study, evaluate and compare these deep learning models in detecting anomalies as signal of intrusions in IoT network to determine their relative suitability in IoT environment.

In this research, anomaly detection models based on LSTM structure and autoencoder structure are developed and tested on IoT-23 dataset. The detection performance of the models is evaluated in terms of accuracy, precision, recall, F1 score, and Area Under the Precision-Recall Curve (AUC-PR). These detection performance metrics reflect the ability of the detection models in detecting anomalies while account for the count false negative and false positive. The results are analyzed to conclude a detection model that performs better using

This is an open access article under the CC BY-NC-SA 4.0 license.

IoT-23 dataset. It is expected that the findings of the research will contribute to the body of knowledge in the field of IoT security, deep learning, and network security. The outcome of the research shall also provide useful information in using deep learning approach to solve cybersecurity problems.

2. Related Work

This section reviews the existing studies that are related to the topic of this research. Ongoing researches of anomaly-based intrusion detection using deep learning algorithm are highlighted.

2.1 Anomaly-based Network Intrusion Detection

Intrusion detection can be done with a few kinds of approaches, mainly host-based or network-based. A network intrusion detection models can monitor, inspect and analyze the network traffic to detect intrusions [3]. The mechanism of intrusion detection can also be either signature-based or anomaly-based. For IoT network with highly diversified types and constraints, they are vulnerable to zero-day attack which cannot be detected by signature-based intrusion detection models [4]. Anomaly-based detection model can learn the normal behavior in a normal network traffic, and identify deviations from the normal pattern, which is often a signal of intrusions.

Anomaly-based network intrusion detection system is a security tool to ensure security within a network. The design of anomaly-based network intrusion detection model has always been an important and popular research area. Zachos [5] proposed a prototype of an anomaly-based intrusion detection system for internet of medical things networks based on machine learning algorithms to provide security solutions to the specific needs of resource limited medical devices. G. Zachos's had pointed out in his paper the lack of existing research on anomaly detection for IoT, especially medical IoT (IoMT) [5]. Koniki [6] also proposed an anomaly-based network intrusion detection system that focuses on detecting DoS attack, probe attack, and root to user attack by using deep learning techniques. However, the authors have admitted that their models require further improvement as their current model architecture achieved a relatively low accuracy. Their studies have shown the potential of anomaly-based intrusion detection approach to detect anomalies in IoT network, and the need for researches on the use of deep learning algorithms as intrusion detection model in IoT.

2.2 Deep Learning Algorithm

Deep learning is a subset of machine learning, which is in the domain of artificial intelligence. Deep learning techniques have wide range of application in various fields because of their capabilities in recognizing pattern. One of the key applications of deep learning algorithm is anomaly detection. By learning the pattern of normal data, detection models based on these deep learning algorithms can identify anomalies when the test data deviates greatly from the pattern they have learnt.

Long Short-Term Memory (LSTM), one of the deep learning algorithms used in this research, is a commonly used algorithm for anomaly detection and prediction. LSTM is an enhanced recurrent neural network with pool of memory cell, and consists of short-term state and long-term state to handle both short-term and long-term information [7]. LSTM architecture consists of three types of memory gates to manage the information flow over long sequences: the forget gate, the input gate, and the output gate. In the context of network anomaly detection, LSTM can learn the pattern in normal network traffic and reconstructs the value of the next data input. The ability of LSTM to capture temporal dependencies had made it a popular choice for projects by researcher or developers. Several studies have shown that LSTM achieving good results as intrusion detection model. D. Ding [8] proposed an intrusion detection system based on a bi-directional LSTM network tailored to detect anomalies the Controller Area Network (CAN) messages within the vehicle internal network, which utilizes sliding window strategy to optimize the time sequence for data input, enabling the model to capture critical temporal dependencies and identify abnormal behavior in the network traffic. This specific study showed Bi-LSTM model achieved excellent result of 100 detection accuracy. Another anomaly-based network intrusion detection system that uses LSTM proposed by R. Koniki [6] shown over 90% accuracy for DoS and Probe attacks, 89% for R2L attacks. These studies affirm the ability of LSTM as potential network intrusion detection model.

The other deep learning algorithm used in this research is autoencoder, which is another commonly used algorithm for anomaly detection. Autoencoder consists of layers of encoders and decoders. The encoding layer encode the data into lower dimension representation, and the decoding layer reconstruct the data. By training an autoencoder model, the reconstruction error can be minimized. This allows the applications of autoencoders in denoising, predicting, and information classifying in datasets [9]. There are many variations of autoencoders. In S. Hore paper [10], variants of autoencoders are compared in detecting malicious activity in a network dataset and the most basic form of autoencoder has shown excellent detection result against other variations in detecting certain attack types.

Both LSTM and autoencoder are popular and well-established models in the field of anomaly detection, each representing a different fundamental approach to identifying anomalies as intrusions in network datasets or real applications. By comparing these two models, this research evaluates two different reconstruction-base

strategies, one that leverages recurrent layers to model temporal dependencies within the data, and one that uses feed-forward layers to model a generalized feature representation.

2.3 Related Existing Studies

Jeelani [11] proposed an intrusion detection system for IoT botnet using machine learning and deep learning algorithms, including Naïve Bayes, Support Vector Machines (SVM), Decision Trees, and deep learning method like Convolutional Neural Networks (CNN). Their model captures the traffic flow into a computer unit and they test ran the machine learning and deep learning models to get the performance and cost of the models. The models are testing on the IoT-23 dataset to obtain the result of performance. Their performance evaluation results for Naïve Bayes, SVM, Decision Tree, CNN are 0.30, 0.69, 0.73 and 0.6935 in terms of testing accuracy.

Another study by Kamalov [12] the autoencoder-based intrusion detection system analyzes the performance of the proposed detection model. The autoencoder-based intrusion detection model is trained in unsupervised manner using benign traffic. The model stacked 3 layers of encoders and 3 layers of decoders, and uses the ReLU activation in all hidden layers along with Adam optimizer. The authors employed the one-class SVM and Robust Covariance (RC) as the benchmark's detection algorithms. The datasets used by Kamalov and his team was the CSE-CIC-IDS2018. The evaluation metrics are accuracy, precision and recall. Under these experimental setups, the authors had achieved 76.71% accuracy, 71.44% precision and 89.81% recall with the autoencoder-based detection model. The numbers have proved that autoencoder-based model perform better overall than the benchmark unsupervised detection algorithm, which is 56.47% accuracy, 48.66% precision, 58.29% recall for one-class SVM, and 56.57% accuracy, 88.82% precision, 52.29% recall for Robust Covariance.

Sah [13]compared the ability of Convolutional Neutral Network (CNN), Long Short-Term Memory (LSTM), Deep Neural Network and Gated Recurrent Unit (GRU) to detect intrusions in a network. The models are tested using NSL-KDD dataset, which is a commonly used benchmark for IDS testing. The NSL-KDD dataset contains balanced network traffic with normal and malicious traffic flows. In short, the DNN has the highest test number among other models, which is 80% accuracy, 73% recall, 90.8% precision, and 81.4% F1 score.

Finally, Jose et. al [14] work on deep learning algorithms performance analysis for intrusion detection in IoT in UNSW-NB15 dataset. The dataset used in their research contains a mix of normal traffic and various types of attack scenarios such as DoS, reconnaissance, and backdoors. The studied models by the Jose's are Decision Tree (DT), Deep Neural Network (DNN) and Convolutional Neural Network (CNN). All models achieve more than 80% accuracy, but result is not as good with other detection metrics. The summary of the related existing studies is shown in Table 1.

Table 1 *Existing Studies Summary*

Work By	Dataset/split ratio	Detection Algorithm	Accuracy	Precision	Recall	F1 Score
Jeelani [11]	IoT-23 / 80:20	Naïve Bayes (NB), SVM, DT, CNN.	NB: 30%, SVM: 69%, DT: 73%, CNN: 69%	NB: 85%, SVM: 60%, DT: 77%	NB: 30%, SVM: 69%, DT: 73%	NB: 30%, SVM: 69%, DT: 73%
Kamalov [12]	CSE-CIC-IDS2018 (80:20)	Autoencoder (AE), SVM, RC	AE: 77%, SVM: 56%, RC: 57%	AE:71%, SVM: 49%, RC: 89%	AE: 90%, SVM: 58%, RC: 52%	NA
Sah [13]	NSL-KDD	LSTM, DNN, CNN, GRU	LSTM: 78%, DNN: 80%, CNN: 77%, GRU: 79%	LSTM: 90%, DNN: 91%, CNN: 90%, GRU: 97%	LSTM: 69%, DNN: 73%, CNN: 67%, GRU: 64%	LSTM: 78%, DNN: 84%, CNN: 77%, GRU: 77%
Jose et. al [14]	UNFW-NB15	DT, DNN, CNN	DT: 80%, DNN: 86%, CNN: 89%	DT: 77%, DNN: 86%, CNN: 84%	DT: 90%, DNN: 73%, CNN: 83%	DT: 83%, DNN: 79%, CNN: 84%

This analysis of existing literature, therefore, points to a clear research gap. While individual studies have demonstrated the use of different deep learning models, there is a lack of a focused comparison between the LSTM and autoencoder architectures when both are used for reconstruction-based anomaly detection on a IoT specific dataset. The use of different datasets and evaluation metrics across the literature makes it difficult for researchers to directly assess the relative strengths and weaknesses of these two popular approaches under the same experimental conditions.

This research may fill this gap. By conducting a direct comparative analysis of an LSTM and an autoencoder model on the IoT-23 dataset. This study will provide a clear performance benchmark. This focused comparison will offer valuable insights into the suitability and effectiveness of each model for anomaly-based intrusion detection in real-world IoT environments.

3. Methodology

This research applies a methodology framework of 5 phases, which follows established practices for comparative studies in machine learning [15]. The flow of the framework is illustrated in Fig. 1. The first phase involves selecting a dataset that can simulate the real-world IoT network environment which contain both normal and anomalous traffics. Second, the phase of data pre-processing involves splitting the dataset into training set, validating set, and testing. This phase also involves preprocessing the data, including feature filtering, data cleansing, and data normalizing to ensure the compatibility for the analysis. The third phase focuses detection models development, which are based on LSTM and autoencoders architecture. This phase includes coding and training the models to detect anomalies in network flows by using a training set that contain only the benign flows, and to validate the trained model by using a validating set. The fourth phase is models' detection performances evaluation, which is where the detection models are evaluated on the testing set using performance metrics including accuracy, precision, recall, and F1-score. The final phase, analyzation of the evaluation results for comparison, includes analyzing the evaluation results to compare the capability of the detection models to detect intrusions in IoT networks.

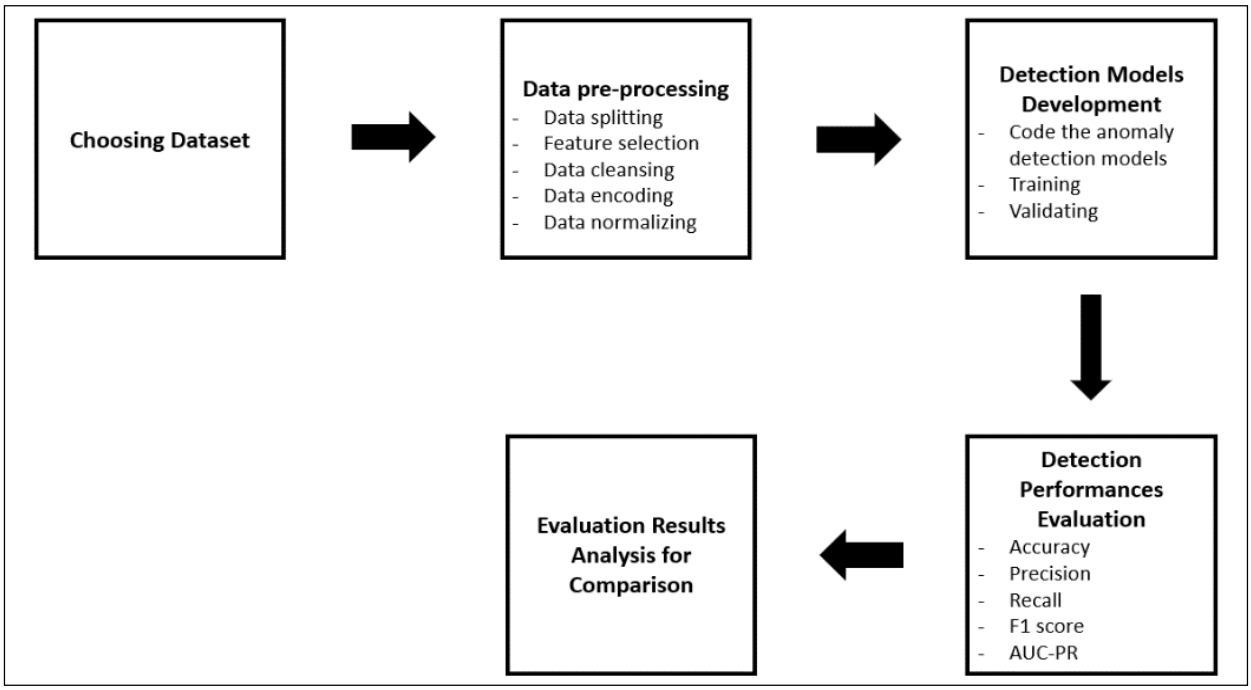

Fig. 1 *Comparative Research Framework*

3.1 Choosing Dataset

The Aposemat IoT-23 dataset, created in the AVAST-AIC lab funded by AVAST, is chosen in this research project to conduct the experiment [16]. The dataset is designed for intrusion detection model benchmarking. It is labeled and it simulates the benign traffic and malware infected traffic in IoT network. The dataset consists of 3 captures of benign traffic and 20 infected traffics, total 21GB with the full dataset, or 8.8GB with the lighter version that contains only the labeled flows without the PCAP files. A capture, also called scenario, is the record of network traffic of IoT devices. The benign captures are collected from 3 real IoT devices, and the malicious captures are collected by executing sample malwares on a Raspberry Pi machine. This raw network traffics are already analyzed and converted into "conn.log.labled" files using a network analysis tool named Zeek. A flow is a combination of packets with similar attributes to represent a complete activity or transaction. IoT-23 has total of 23 columns, and their description is shown in Table 2.

Table 2 *IoT-23 Dataset Columns Description [16]*

Column	Types	Description
ts	time	Timestamp of the flow
uid	string	Unique identifier of the flow
id.orig_h	addr	Source IP
id.orig_p	port	Source port
id.resp_h	addr	Destination IP
id.resp_p	port	Destination port
proto	enum	TCP, UDP or ICMP
service	string	services recognized by Zeek: HTTP, DNS, TELNET, SSL, SSH, IRC
duration	interval	Total duration of the flow
orig_bytes	count	Number of payload bytes sent by originator
resp_bytes	count	Number of payload bytes sent by the responder
conn_state	string	Connection state: S0, S1, SF, REJ, S2, S3, RSTO, RSTR, RSTOS0, RSTRH, SH, SHR, OTH
local_orig	bool	If locally originated connection: T. If remotely originated connection: F
local_resp	bool	If locally responded connection: T. If remote responded connection: F
missed_bytes	count	Packet loss
history	string	A sequence of characters made up from a fixed character set, each representing a connection in history state. Character is capitalized if made by responder, lower case if made by originator [17]
orig_pkts	count	Number of packets sent by originator
orig_ip_bytes	count	Number of IP level bytes sent by the originator
resp_pkts	count	Number of packets sent by the responder
resp_ip_bytes	count	Number of IP level bytes sent by the responder
tunnel_parents	set[string]	Log the encapsulating parent connections if connections through tunnel
label	string	Label the data as “Benign”, or “Malicious”
detailed-label	string	Further classifications of “Malicious” label

3.2 Data Pre-processing

Data pre-processing involves restructuring and converting the data from the dataset into a representation that aligns with the context of the study. The steps include data splitting, feature selection, data cleaning, data encoding, and data normalization. This makes sure the processed data is ready for detection models input. In this research, the pre-processing of the dataset is done using Python, with Pandas library, scikit-learn, and NumPy. Pandas is a library that is design for data analysis data manipulation, scikit-learn is a library that provides data pre-processing tools in the context of machine learning, and NumPy is a powerful array library.

3.2.1 Data Splitting

In order to prevent data leakage, it is important to split the dataset before applying any pre-processing technique [18]. The common practice is to split the dataset into training set, validating set, and testing set by ratio 80:10:10. Since the detection models are only trained using the benign flows, all malicious flows in the training set is discarded for unsupervised training. The splitting method used is stratified sampling, which ensures the class distribution of benign and malicious is the same across all samples. In Python, “scikit-learn” provides a convenient method to for data splitting, which is the “train_test_split()” of the “model_selection” module.

3.2.2 Features Selection

Feature selection is an approach of feature reduction in minimizing data redundancy and excluding irrelevant features so the model can learn patterns more effectively [19]. In this research, the features that are redundant, null-value dominant, and have insignificant purpose in anomaly detection are dropped.

First, all features that serve as identifiers are dropped including "uid" and "ts". Then, the host identifiers "id.orig_h", and "id.resp_h" are also dropped. "orig_ip_bytes" is a multiplication of "orig_pkts", and "resp_ip_bytes" is a multiple of "resp_pkts". These pairs are directly correlated, hence one of the features in each pair are dropped. The features that are related to port number do not have significant information for identifying anomalies in network, hence features that relate to port number such as "id.orig_p" and "id.resp_p" are also dropped [17]. Finally, features that have mostly 0 value or null entries like "local_orig", "local_resp", "tunnel_parents" are dropped as well as the "detailed-label" feature which is irrelevant to the study.

The final features remained are "proto", "service", "duration", "orig_bytes", "resp_bytes", "conn_state", "missed_bytes", "orig_ip_bytes", "resp_ip_bytes", and "history". The "label" column is used as the target to differentiate between benign flows and malicious flows. Among the remaining feature, the numerical features are "duration", "orig_bytes", "resp_bytes", "missed_bytes", "orig_ip_bytes", and "resp_ip_bytes". Categorical features are "proto", "service", and "conn_state". "history" is a string sequence made up from a fixed range of character set. The "label" column is the target, which indicates the class of the flow, whether "Benign" or "Malicious".

3.2.3 Data Cleansing

Data cleaning is the process of handling missing values and inconsistencies in formatting. In this data pre-processing step, the data will be undergoing deduplication to remove redundant records in the dataset. The purpose of deduplication is to prevent model overfitting to the repeated data, causing poor generalization on unseen data [20]. To prevent data leakage, statistics such as mean and mode in the training set are used to replace missing values in all three sets. For missing values in numerical columns, the meaning in the training set is used as replacement. Similar action is taken for missing values in categorical columns except "service", the mode in the training set is used to replace the missing values. The missing values in "service" do not get replaced with anything because it carries the information of the fact that there is no specific service recognized.

3.2.4 Data Encoding

Data encoding is the process of encoding features into numerical format that is readable by the detection models. One of the most common encoding techniques is one-hot encoding. It creates one-hot encoding features which are Boolean type for each possible category, and the value will be 1 for data points in that specific category with all 0s in the others. On the other hand, the "history" feature is encoded into aggregated features, which features correspond to each character including its capitalization in the character set are created: 's', 'h', 'a', 'd', 'f', 'r', 'c', 'g', 't', 'w', 'i', 'q', '^'. The values of these aggregated features will be the occurrences of each character in the original string.

3.2.5 Data Normalization

Data normalization includes rescaling the numerical features into a fixed range. This can effectively prevent the arithmetic effect when values in some columns are significant on larger scale. A paper suggests that min-max normalization method can improve the detection performance of the detection models, and allows the models to work better in terms of precision [21]. Therefore, the numeric features will be normalized using the min-max scaling. The "MinMaxScaler" object class from the scikit-learn library will be used to complete the task. Min-max scaling involves the use of the minimum value and maximum value in the feature as scaling references. All numeric features in training set, validating set, and testing set are scaled based on the statistics in the training set.

3.3 Detection Models Development

The anomaly-based intrusion detection models that are being compared in this research are the LSTM model and the autoencoder model. Based on the research framework, detection models' development includes coding of the model structure, training, and validating. The coding for the structure of the detection models is done using Python programming language with the PyTorch framework.

3.3.1 LSTM-based Anomaly Detection Model

A LSTM model is made up of basic LSTM units, where the units are made up of series of circuit combinations. The structure for the LSTM used in this research is a 2 layers LSTM model with 64 LSTM units in each layer. For each input, the LSTM outputs the corresponding reconstruction. The structure of the LSTM-based detection model in this research is shown in Fig. 2.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1037-1050

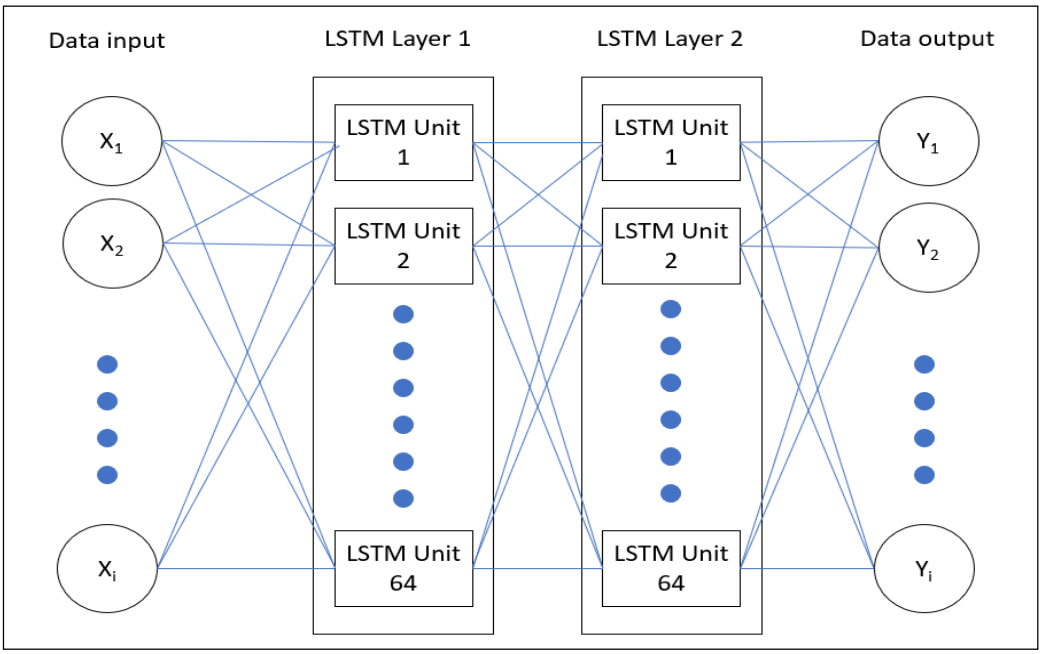

Fig. 2 *2 Layers LSTM Model Structure [22]*

3.3.2 Autoencoder-based Anomaly Detection Model

An autoencoder model is made up of a series of encoder and decoder. The autoencoder used in this research is a 2 hidden layers symmetrical structure. It has 64 hidden units at the first encoding layer and at the last decoding layer, the layer after has 32 hidden units, and the bottleneck is 16 hidden units. The autoencoder model in this research will also use the ReLU activation function in each layer to make sure the input to the autoencoder model is a non-negative input. The structure of the autoencoder-based detection model is shown in Fig. 3.

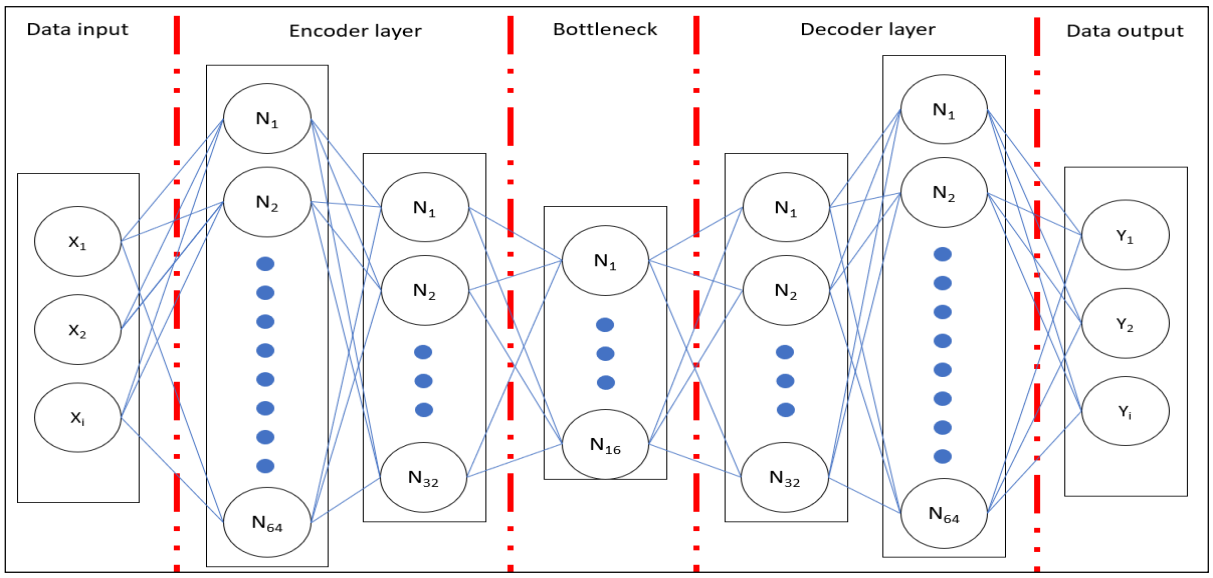

Fig. 3 *2 Hidden Layers Autoencoder Model Structure [23]*

3.3.3 Detection Models Training

Loss function, which is a mathematical function to quantify the performance, is Mean Squared Error (MSE) for both detection models. It is the mean of squared differences between the models output and the data input. During training, the main goal is to minimize this MSE, meaning that the reconstructed data should be very much similar to the actual data [24]. The optimizer used for training is the Adam optimizer, which dynamically adjusts the models' internal parameters to minimize the MSE. The models are trained using a batch size of 32, a learning rate of 0.001, and for 50 epochs, allowing sufficient iterations for the models to learn meaningful patterns from the data.

3.3.4 Detection Models Validation

The main goal of the validation step is to determine the optimal threshold for the detection task. A threshold is the maximum MSE of the output by the detection models that can be accepted as a benign data. The detection models are trained to reconstruct the benign data with minimal MSE. Therefore, an output that exceeds the threshold is deemed as malicious as it suggests that the detection models have not seen it during training.

The selection of an optimal threshold is crucial as it directly affects the detection performance. The process of determining the best threshold is called threshold tuning. In this research, the approach for threshold tuning is by plotting a precision-recall curve using the validating set. This is done by using the "precision_recall_curve()" method in "sklearn.metrics" module, which uses every possible MSE in the validating set to plot the precision-recall curve. On the precision-recall curve, the optimal threshold is determined by finding the plot that has that best F1 score, which balances between precision and recall.

3.4 Detection Performances Evaluation

The criteria to classify whether an input is 'benign' or 'malicious' is based on the MSE between the algorithms output and the original input. During training, the detection models had learnt output a reconstruction of 'benign' input. Therefore, the MSE of a 'benign' input should be low, and if an input results in a MSE exceeding the MSE threshold in the validation step, the input is classified as anomaly. Since the task is anomaly detection, an anomaly is referred as positive, and a benign data is referred as negative.

The evaluation is done using the 10% testing set. The "label" field in the dataset will be used as the ground truth to compare the models' detection result with the actual class of the data. Each detection is classified as either True Positive (TP), True Negatives (TN), False Positive (FP), or False Negatives (FN). These counts are used to calculate the key evaluation metrics which are accuracy, precision, recall, F1 score, and AUC-PR.

In Python, built-in functions to calculate these metrics are provided by the "sklearn.metrics" module. The equations for accuracy, precision, recall, and F1 score [25] are shown in Eq. 1 to 4 respectively. Whereas the AUC-PR is calculated using trapezoidal rule by the "auc()" method from "sklearn.metrics" module.

$$Accuracy = \frac{TP + TN}{Total\ Count} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1\ Score = \frac{2\ \times Precision\ \times Recall}{Precision + Recall} \tag{4}$$

3.5 Detection Performances Comparison

The final step of the methodology involves analyzing the evaluation results of the LSTM-based model and the autoencoder-based model in detecting anomalies as intrusion in the IoT-23 dataset. The evaluation metric mentioned in previous sections, which are accuracy, precision, recall and F1 score are the main focuses of the analysis. Each metrics assess the models in different aspects of detection. Accuracy evaluates the ability of the models in identifying anomalies correctly in the network flows, precision evaluates the ability of the models to avoid flagging benign flows as anomalous, recall evaluates the sensitivity of the models, and F1 score providing a balance that accounts both the precision and recall of the models.

Additionally, the Area Under the Precision-Recall Curve (AUC-PR) is included as a complementary metric to further evaluate the detection models' performances. AUC-PR provides a scalar value between 0 and 1. A value of 1.0 indicates perfect generalization across all thresholds, a value around 0.5 suggests the detections are only about as good as random guessing, a value lower the 0.5 suggests the detections are worse than random guessing [26].

The outcome of the analysis can show the strengths and weaknesses of each detection model. A more suitable model to detect intrusion in IoT networks is determined.

4. Result and Discussion

This section presents the results obtained from the implementation of the research and discusses the results to achieve the research objectives.

4.1 Pre-processed Dataset Summary

The data pre-processing steps done on the IoT-23 dataset include dataset splitting, feature selection, data cleansing, data encoding, and data normalization. The total number of features extract from the dataset is 10, which are "proto", "service", "duration", "orig_bytes", "resp_bytes", "conn_state", "missed_bytes", "orig_ip_bytes", "resp_ip_bytes", and "history". Some of these features have been encoded and derived into additional columns, resulting in total of 50 features to be fed as input to the detection models. After the data pre-processing steps, the number of records in the training set, validating set, and testing set are around 300 thousand. The summary of the distribution is presented in Table 3.

Table 3 *IoT-23 Dataset Columns Description*

Training Set		Validating Set		Testing Set	
Benign	Malicious	Benign	Malicious	Benign	Malicious
101214	0	41548	234034	41793	233999

4.2 Detection Models Development

The models are trained with an initialization of parameters using a random seed of 88 in "torch.manual_seed()". Based on the threshold tuning during model validation step, the optimal threshold is found to be 3.332e-05 for LSTM model and 9.409e-06 for autoencoder model. These thresholds are used in evaluation for both models respectively.

4.3 Detection Performances Evaluation Results

The evaluations of the detection performance for the LSTM model and the autoencoder are both excellent and competitive. For accuracy, the LSTM model achieved a value of 0.881 and the autoencoder model achieved a value of 0.853. For precision, the LSTM model achieved a value of 0.986 and the autoencoder model achieved a value of 0.852. For recall, the LSTM model achieved a value of 0.873 and the autoencoder model achieved a value of 1.000. For F1 score, the LSTM model achieved a value of 0.926 and the autoencoder model achieved a value of 0.920. For AUC-PR, the LSTM model achieved a value of 0.982 and the autoencoder model achieved a value of 0.973. The summary of the evaluation results is presented in Table 4.

Table 4 *Summary of Evaluation Results*

Model	Accuracy	Precision	Recall	F1 Score	AUC-PR
LSTM	0.881	0.986	0.873	0.926	0.982
Autoencoder	0.853	0.852	1.000	0.920	0.973

4.4 Detection Performances Evaluation Discussion and Comparison

The discussion provides a detailed analysis and comparison of the detection performances of the LSTM and Autoencoder models based on key evaluation metrics. The strengths and limitations of each model are highlighted. The comparison aims to determine which model performs better in identifying anomalies as intrusion while maintaining a balance between detection sensitivity and reliability. To support this analysis, a comprehensive bar chart visualization of the evaluation metrics is presented in Fig. 4. The blue bars represent the LSTM model and the orange bars represent the autoencoder model. The y-axis displays zoomed-in scores for each metric to highlight the differences between the two models. The bar charts clearly show that one model outperforms the other for precision and recall.

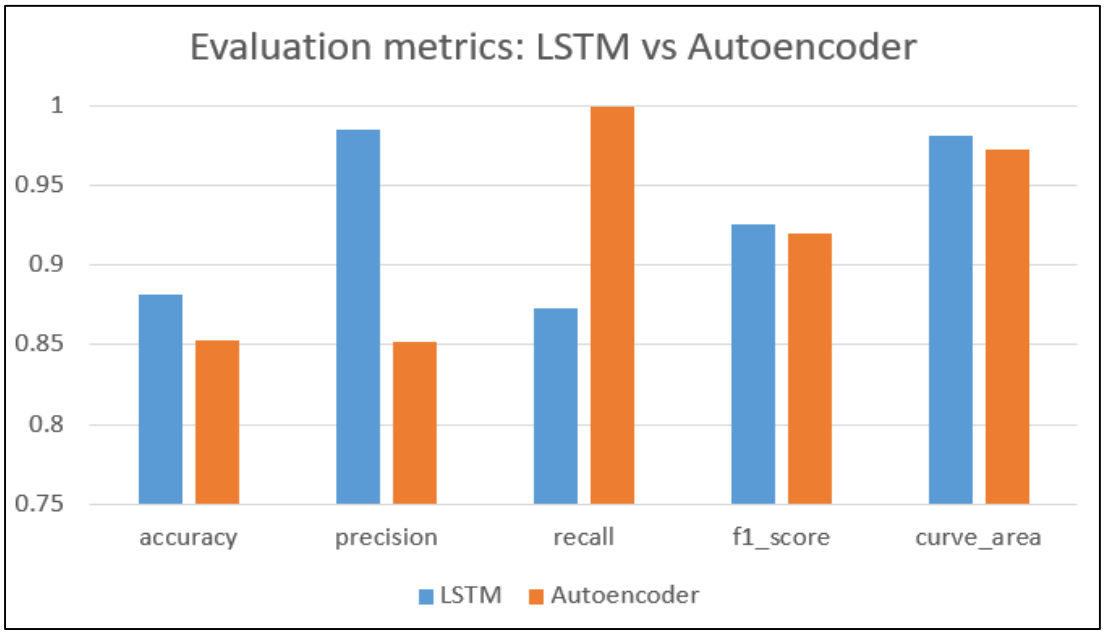

Fig. 4 *Bar Chart of the Evaluation Metrics*

4.4.1 Accuracy Comparison

Accuracy is a fundamental metric that reflects the overall correctness of a model's predictions, which is the ratio of true positives and true negatives to the total number of instances. The LSTM model achieved an accuracy of 0.881, while the autoencoder model recorded an accuracy of 0.853. Both models have been recorded with a relatively high accuracy. However, the difference of accuracy between the models is notable, with the LSTM model demonstrating a better generalization of the testing set.

In the context of intrusion detection for IoT network, higher accuracy suggests that the model is more reliable in identifying both benign traffic and malicious traffic. Based on the observation of the evaluation of accuracy, the LSTM model is a more favorable option compared to the autoencoder model as it offers a better overall prediction performance.

4.4.2 Precision Comparison

Precision measures the proportion of identified malicious flows to the total number of malicious flows in the dataset. A higher precision indicates the model generates fewer false alarms, which is crucial in reducing unnecessary security alerts in a real-world intrusion detection system.

Based on the evaluation result, the LSTM model has significantly better precision than the autoencoder. The LSTM model recorded a precision of 0.986, while autoencoder model recorded a precision of 0.852. This indicates that the LSTM model rarely misclassifies benign traffic as malicious.

For IoT environments, where devices are typically resource-constrained and manual operations by human are intended to be limited, high precision is important to avoid wasting resources on false alarms and to maintain operational efficiency. The evaluation suggests that in terms of precision, LSTM is a much better choice as intrusion detection model for IoT network. On the other hand, although the autoencoder model has reasonable precision, it is more prone to false positive, alarming benign traffics as malicious. Overtime, it might lead to higher alert fatigue. This comparison highlights the LSTM model's superior capability in detecting malicious traffic, without triggering false alarm, making it more practical for real-time intrusion detection model.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 1037-1050

4.4.3 Recall Comparison

In contrast to the precision comparison, the autoencoder is superior to the LSTM in terms of recall. Based on the evaluation result, the LSTM model recorded a recall of 0.873, while autoencoder model recorded a precision of 1.000. This indicates that the autoencoder model is perfect at detecting all malicious flows, without any false negatives.

For IoT environments, although efficiency is often favored, certain applications or systems may security to be prioritized over performance. For example, in a critical infrastructure where any missed malicious traffic could lead to significant harm, it is crucial to achieve a perfect recall. Therefore, this evaluation suggests that autoencoder is much suitable for a critical IoT network as it ensure all intrusions can be detected, with the trade-off of false alarm. Likewise, while the LSTM model having a reasonable recall, it still misses a portion of the malicious flows, making it less secure especially when successful detection of malicious traffic is highly required.

This comparison highlights the autoencoder model's superior capability in detecting all malicious traffic without a miss, making it a much cautious choice for when high security is the main requirement.

4.4.4 F1 Score Comparison

F1 score is a balanced metric that takes into the account of both precision and recall. It is useful at providing insightful performance evaluation in imbalanced detection tasks like intrusion detection. Based on the evaluation result, the LSTM model recorded a F1 score of 0.926, while the autoencoder model recorded a slightly lower F1 score of 0.920. Both models demonstrated a good balance between precision and recall with their high relative F1 score. Despite the good balanced performance suggested by the models' F1 score, it is important that both models achieved this result with different trade-off. LSTM model has exceptional precision but relatively lower in its recall. Autoencoder model on the other hand, achieves a perfect recall but has relatively lower precision.

This comparison shows that the LSTM model is more balanced, and better overall detection models. It also shows that LSTM model is more useful in most of the practical case where both false alarms and missed detections need to be balanced. In contrast, while the autoencoder model is still highly competitive, it is favored only when false negatives need to be minimum.

4.4.5 AUC-PR Comparison

The Area Under the Precision-Recall Curve (AUC-PR) is a robust performance metric that captures the trade-off between precision and recall across various threshold values. It provides a comprehensive view of a model's ability to balance false positives and false negatives, especially important in an imbalance scenario such as intrusion detection.

The LSTM model achieved an AUC-PR of 0.982, indicating a strong and consistent performance in distinguishing malicious traffic across a wide range of thresholds. This high value suggests that the LSTM model is able to maintain both high precision and recall when the threshold shifts. Such stability makes the model more reliable and adaptable in the real world where optimal thresholds may shift over time due to changing network behavior or traffic patterns. The precision-recall curve of the LSTM model is presented in Fig. 5.

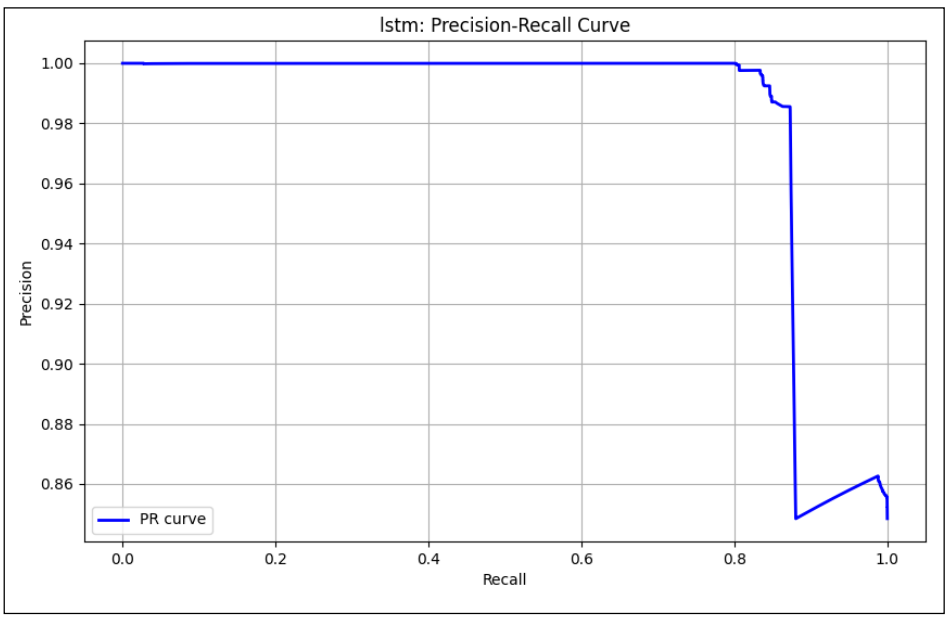

Fig. 5 *Precision-Recall Curve of LSTM Model in Evaluation*

In comparison, the autoencoder model recorded a slightly lower AUC-PR of 0.973 Although this is still a strong performance, its lower AUC-PR suggests that its precision-recall balance may be more sensitive to threshold changes. Notably, the autoencoder model achieved perfect recall at the chosen optimal threshold, which is ideal for detecting all intrusions when the system is critical. But when compared with the LSTM model, the lower AUC-PR suggests that its performance could degrade more rapidly outside the optimal threshold range. The precision-recall curve of the autoencoder model is presented in Fig. 6.

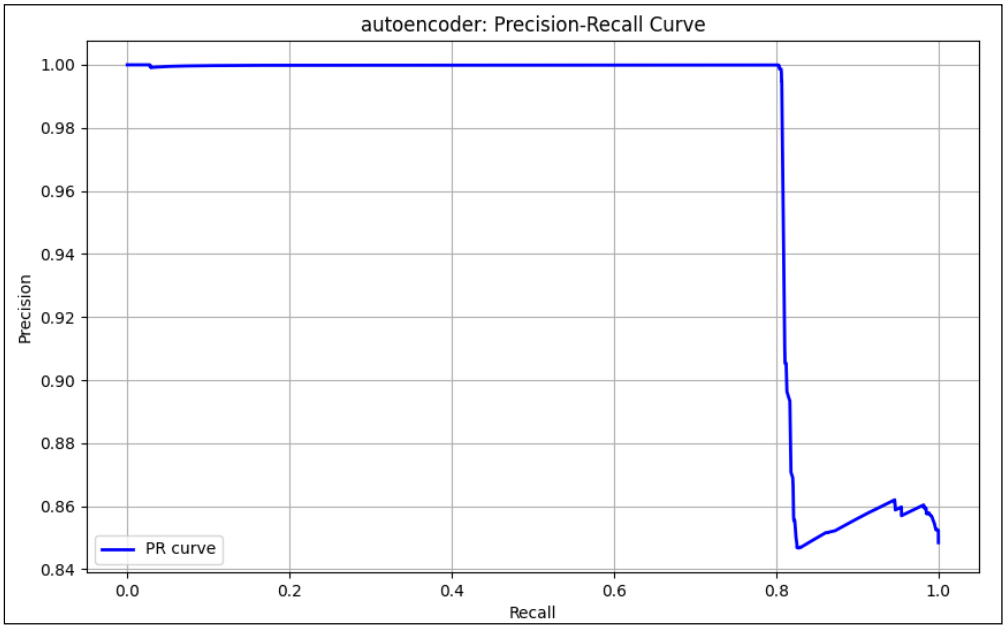

Fig. 6 *Precision-Recall Curve of Autoencoder Model in Evaluation*

Overall, the comparison in terms of AUC-PR highlights that while both models perform well, the LSTM model demonstrates an edge in terms of adaptability and consistent detection performance across varying thresholds. This makes it a more robust choice for real-world deployment in IoT network intrusion detection systems, where flexibility and reliability over time are critical.

5. Conclusion

The evaluation results showed the LSTM model achieved an accuracy of 0.878, precision of 0.981, recall of 0.873, F1 score of 0.926, and AUC-PR of 0.982, excelling in precision and a balanced performance to minimize false positives. In comparison, the Autoencoder model recorded an accuracy of 0.853, precision of 0.852, recall of 1.000, F1 score of 0.920, and AUC-PR of 0.973, achieving perfect recall to ensure no malicious flows were missed. The LSTM's higher AUC-PR indicates better adaptability across varying thresholds, making it a more robust option in dynamic environment.

Overall, the LSTM model shows a more balanced and stable performance for intrusion detection in IoT network, making it suitable for most of the general use-cases. Its superior AUC-PR also suggests that it adapts better when the environment changes. On the other hand, the autoencoder model had also achieved a competitive metrics with a perfect recall of 1.000, making it more favorable in high security context where the priority is to detect all malicious flow with trade-off of false alarms.

These findings highlight the strengths and weaknesses of each model. It also suggests that model selection should consider the specific needs of the deployment environment. For future research, further study could explore a hybrid model structure where the LSTM and autoencoder are combined to provide a more robust performance. Furthermore, the evaluation can be extended to more diverse IoT datasets to train the models in a more diverse environment. Finally, the optimization for resource-efficiency is also a good study that extend the foundation of this research, as IoT devices are usually constrained with resources.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *S. C. Lim, K. M. Mohamad;* ***data collection:*** *S. C. Lim, K. M. Mohamad;* ***analysis and interpretation of results:*** *S. C. Lim, K. M. Mohamad;* ***draft manuscript preparation:*** *S. C. Lim, K. M. Mohamad. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] M. Tawfik, "Optimized intrusion detection in IoT and fog computing using ensemble learning and advanced feature selection," *PLoS One*, vol. 19, no. 8, pp. 1–41, Dec. 2024, doi: 10.1371/journal.pone.0304082.

[2] Z. Ali, H. Ali, and M. Badawy, "Internet of Things (IoT): Definitions, Challenges, and Recent Research Directions," *Int J Comput Appl*, vol. 128, pp. 975–8887, Dec. 2015.

[3] S. K. Kodali and C. H. Muntean, "An Investigation into Deep Learning Based Network Intrusion Detection System for IoT Systems," in *2021 IEEE International Conference on Data Science and Computer Application (ICDSCA)*, 2021, pp. 374–377. doi: 10.1109/ICDSCA53499.2021.9650111.

[4] D. Nair and N. Mhavan, "Augmenting Cybersecurity: A Survey of Intrusion Detection Systems in Combating Zero-day Vulnerabilities," 2023, pp. 129–153. doi: 10.1108/S1569-37592023000110A007.

[5] G. Zachos, G. Mantas, I. Essop, K. Porfyrakis, J. C. Ribeiro, and J. Rodriguez, "Prototyping an Anomaly-Based Intrusion Detection System for Internet of Medical Things Networks," in *2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2022, pp. 179–183. doi: 10.1109/CAMAD55695.2022.9966912.

[6] R. Koniki, M. D. Ampapurapu, and P. K. Kollu, "An Anomaly Based Network Intrusion Detection System Using LSTM and GRU," in *2022 International Conference on Electronic Systems and Intelligent Computing (ICESIC)*, 2022, pp. 79–84. doi: 10.1109/ICESIC53714.2022.9783500.

[7] S. M. Al-Selwi *et al.*, "RNN-LSTM: From applications to modeling techniques and beyond—Systematic review," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 5, p. 102068, 2024, doi: https://doi.org/10.1016/j.jksuci.2024.102068.

[8] D. Ding, L. Zhu, J. Xie, and J. Lin, "In-Vehicle Network Intrusion Detection System Based on Bi-LSTM," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2022, pp. 580–583. doi: 10.1109/ICSP54964.2022.9778620.

[9] P. Li, Y. Pei, and J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Appl Soft Comput*, vol. 138, p. 110176, 2023, doi: https://doi.org/10.1016/j.asoc.2023.110176.

[10] S. Hore, Q. H. Nguyen, Y. Xu, A. Shah, N. D. Bastian, and T. Le, "Empirical Evaluation of Autoencoder Models for Anomaly Detection in Packet-based NIDS," in *2023 IEEE Conference on Dependable and Secure Computing (DSC)*, 2023, pp. 1–8. doi: 10.1109/DSC61021.2023.10354098.

[11] F. Jeelani, D. S. Rai, A. Maithani, and S. Gupta, "The Detection of IoT Botnet using Machine Learning on IoT-23 Dataset," in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, 2022, pp. 634–639. doi: 10.1109/ICIPTM54933.2022.9754187.

[12] F. Kamalov, R. Zgheib, H. H. Leung, A. Al-Gindy, and S. Moussa, "Autoencoder-based Intrusion Detection System," in *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, 2021, pp. 1–5. doi: 10.1109/ICEET53442.2021.9659562.

[13] N. K. Sah, M. Kolli, K. P. Dharmaraj, and H. N. Vishwas, "Comparative Deep Learning Approach for Intrusion Detection," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10724847.

[14] J. Jose and D. V Jose, "Performance Analysis of Deep Learning Algorithms for Intrusion Detection in IoT," in *2021 International Conference on Communication, Control and Information Sciences (ICCISc)*, 2021, pp. 1–6. doi: 10.1109/ICCISc52257.2021.9484979.

[15] Ainurrochman *et al.*, "Ensemble Methods Classifier Comparison for Anomaly Based Intrusion Detection System on CIDDS-002 Dataset," in *2021 13th International Conference on Information & Communication Technology and System (ICTS)*, 2021, pp. 62–67. doi: 10.1109/ICTS52701.2021.9608714.

[16] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic ," May 2021, *Zenodo*. doi: 10.5281/zenodo.4743746.

[17] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul, "Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset," *IEEE Access*, vol. 10, pp. 6430–6441, 2022, doi: 10.1109/ACCESS.2021.3140015.

[18] A. R. Gad, A. A. Nashat, and T. M. Barkat, "Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset," *IEEE Access*, vol. 9, pp. 142206–142217, 2021, doi: 10.1109/ACCESS.2021.3120626.

[19] Y.-X. Meng, "The practice on using machine learning for network anomaly intrusion detection," in *2011 International Conference on Machine Learning and Cybernetics*, 2011, pp. 576–581. doi: 10.1109/ICMLC.2011.6016798.

[20] E. Slyman, S. Lee, S. Cohen, and K. Kafle, "FairDeDup: Detecting and Mitigating Vision-Language Fairness Disparities in Semantic Dataset Deduplication," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 13905–13916. doi: 10.1109/CVPR52733.2024.01319.

[21] P. V Pandit, S. Bhushan, and P. V Waje, "Implementation of Intrusion Detection System Using Various Machine Learning Approaches with Ensemble learning," in *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, 2023, pp. 468–472. doi: 10.1109/InCACCT57535.2023.10141704.

[22] Z. Chang, W. Yuan, and K. Huang, "Remaining useful life prediction for rolling bearings using multi-layer grid search and LSTM," *Computers and Electrical Engineering*, vol. 101, p. 108083, 2022, doi: https://doi.org/10.1016/j.compeleceng.2022.108083.

[23] D. Azzalini, B. Flammini, C. A. Emanuele, A. Guadagno, E. Ragaini, and F. Amigoni, "An empirical evaluation of deep autoencoders for anomaly detection in the electricity consumption of buildings," *Energy Build*, vol. 327, p. 115069, 2025, doi: https://doi.org/10.1016/j.enbuild.2024.115069.

[24] Y. Lin, J. Wang, and M. Cui, "Reconstruction of Power System Measurements Based on Enhanced Denoising Autoencoder," in *2019 IEEE Power & Energy Society General Meeting (PESGM)*, 2019, pp. 1–5. doi: 10.1109/PESGM40551.2019.8973925.

[25] J.-O. Palacio-Niño and F. Berzal, "Evaluation Metrics for Unsupervised Learning Algorithms," May 2019, doi: 10.48550/arXiv.1905.05667.

[26] S. A. Khan and Z. A. Rana, "Evaluating Performance of Software Defect Prediction Models Using Area Under Precision-Recall Curve (AUC-PR)," in *2019 2nd International Conference on Advancements in Computational Sciences (ICACS)*, 2019, pp. 1–6. doi: 10.23919/ICACS.2019.8689135.