Journal of Theoretical and Applied Information Technology
15th March 2023. Vol.101. No 5

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DAE-BILSTM: A FOG-BASED INTRUSION DETECTION MODEL USING DEEP LEARNING FOR IOT

IBRAHIM MOHSEN SELIM[1] , ROWAYDA A. SADEK[2]
[1,2]Information Technology Department, Faculty of Computers and Artificial Intelligence, Helwan University, Cairo, Egypt
E-mail: [1]ibrahiselim@fci.helwan.edu.eg, [2]rowayda_sadek@fci.helwan.edu.eg

ABSTRACT

Fog computing efficiently brings services to the edge network because it facilitates processing, communication, and storage to be closer to the edge devices. Fog computing is principally used for the Internet of Things (IoT) instead of cloud computing as an ideal option to reduce latency, especially for time-sensitive applications. Many different security challenges and cyber-attacks have emerged in the fog layer, rendering typical defense solutions ineffective. The Network Intrusion Detection System (NIDS) is a security system that uses various advanced learning approaches to detect and predict various attacks. IoT devices lack the computing power and energy needed for such, hence, IDS must be at the fog node to monitor data coming from different sources, such as IoT devices or neighboring fog nodes. To identify cyber-attacks and malicious states in IoT network traffic, this paper proposes an intrusion detection model named DAE-BiLSTM based on deep learning (DL) in the fog layer. The DAE-BiLSTM model employs a Deep AutoEncoder (DAE) in conjunction with Bidirectional Long Short-Term Memory (BiLSTM). Many existing state-of-the-art studies used the NSL-KDD dataset, which is now out of date and does not include new IoT cyber-attacks. The results of these studies also need to be improved by using advanced DL techniques to be effective in detecting current cyber-attacks. So the network-based ToN_IoT dataset is used for the training and testing of the DAE-BiLSTM model. The ToN_IoT dataset comprises new and better-suited cyber-attacks for IoT. The DAE-BiLSTM model obtained a high accuracy rate of 99.7%, 99.4% for precision, 99.7% for recall, and 99.5% for the F1-score in classifying normal traffic data from attacked traffic data in the fog layer.

Keywords: *Internet of Things (IoT), Fog Computing, Intrusion Detection System (IDS), ToN_IoT dataset, Deep Learning (DL)*

1. INTRODUCTION

The Internet of Things (IoT) has led to the large-scale and high-speed development of many systems in various fields of life. IoT is used in smart industrial systems, healthcare systems, smart homes, smart vehicles, and more. Given the nature of IoT, which relies on the exchange of huge amounts of data between large numbers of devices, the use of the cloud is essential. The cloud provides an infrastructure that enables users to acquire computing and storage resources without suffering high design and acquisition costs as needed [1]. However, dealing with the cloud layer directly is now not suitable for latency-sensitive applications. The most common architecture of fog computing can be seen in Figure 1. Fog computing is distributed computing that offers similar services as cloud computing but on a smaller geographic scale [2].

Fog computing between the cloud and the IoT is designed to solve the shortcomings and issues in the cloud such as delays in response time, lack of location awareness, lack of mobility support, as well as lack of geographic distribution..

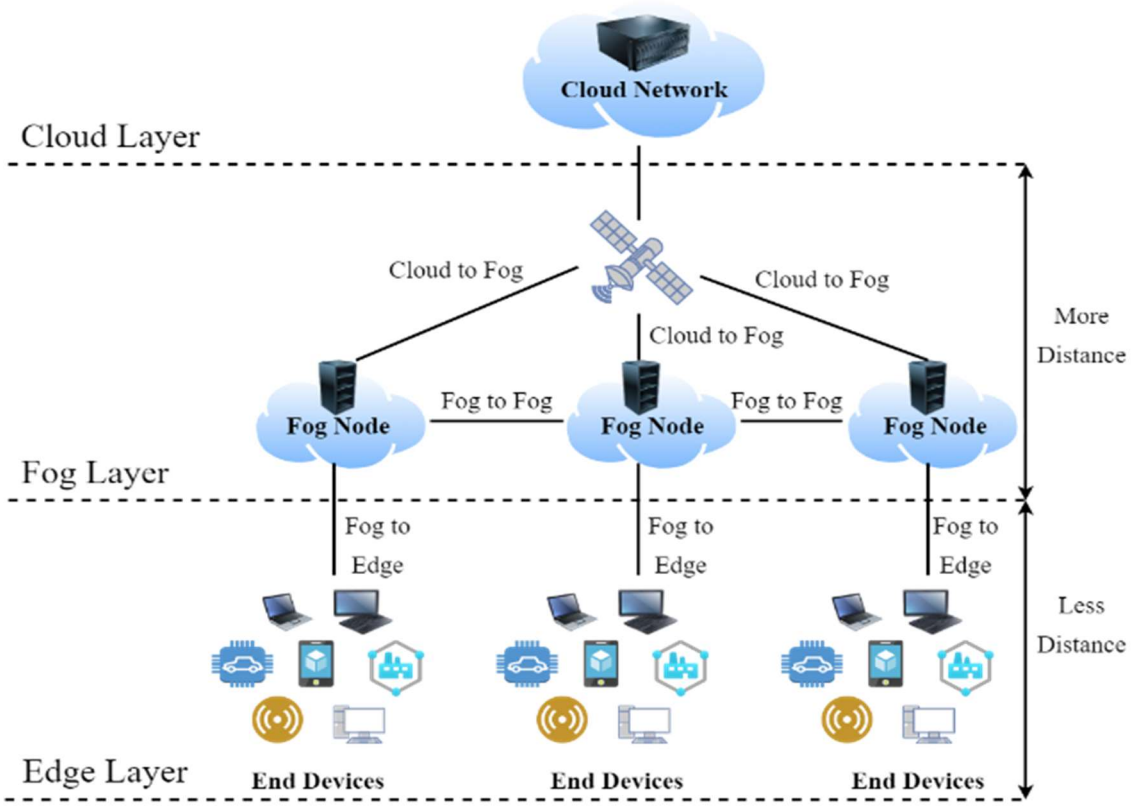

Figure 1: Fog Computing's Hierarchical Architecture

Because fog nodes are placed between the cloud and IoT, they are vulnerable to many cyber threats due to the presence of many insecure devices and the inability to protect the network very efficiently. The Fog Network is a connected network that handles thousands of IoT devices. Any of these devices may only generate small amounts of data, but when data is received from so many devices, it becomes more difficult to secure while it is being processed and more vulnerable to hacking. Attackers also use these vulnerabilities to hack fog devices, obtain sensitive data, and compromise fog services. In addition, there are new attacks specific to their new location at the edge of the network, such as zero-day attacks, flooding attacks, abuse of service, advanced persistent threats, port scanning attacks, backdoor attacks, and user-to-root attacks [3]. Fog layer resources need to be secured and protected from various threats to preserve them. For example, a distributed denial of service (DDoS) attack targets a fog node, which is one of the most powerful flood attacks. Fog has fewer resources than the cloud. These resources are consumed faster, and then the network performance will decrease, which will deprive users of its services. Therefore, there was a need to monitor the packets coming into the fog network and predict attacks before they happened, so intrusion detection systems (IDSs) were used.

Intrusion Detection System (IDS) is widely applied in many systems because it is a valuable solution to reduce malicious attacks by keeping track of network traffic and analyzing the behavior of devices. IDS employs a variety of techniques, including machine learning (ML) and deep learning (DL), to predict and discover attacks before they happen [4]. There are currently three layers in the network architecture: cloud, fog, and edge. IDS can be placed inside the cloud to perform this attack analysis but it takes a lot of time, from hours to days. That is not commensurate with the response speed required for latency-sensitive applications. Therefore, it is preferable to use IDS inside a fog node rather than directly in the cloud to detect any unwanted behavior faster and to monitor and analyze data and files. From the above, we understand that latency is one of the main reasons why we have IDS inside fog nodes. IDS inside a fog node reduces latency between edge devices and cloud servers. Thus, normal network traffic data coming from high-end devices is analyzed and monitored, and malicious attacks are detected in the fog layer.

In the proposed work, a detailed, effective intrusion detection model named DAE-BiLSTM for detecting cyber-attacks via IoT in fog computing is presented. Most of the current research [5], [6], and [7] is evaluated using an outdated NSL-KDD [8] dataset that is not useful for the current IoT environment. In this context, the new ToN_IoT [9] dataset was used to evaluate how effectively the proposed work was done. The ToN_IoT dataset contains all new threats to the fog layer. The popular NSL-KDD dataset was also used in this research for comparison with others to ensure the efficiency of this work. Most of the previous work uses learning techniques that have become weaker at detecting attacks and take more time, and this is no longer suitable for the fog layer. The DAE-BiLSTM uses modern learning techniques to provide intrusion detection within a fog node to reduce latency and increase the accuracy of attack detection. The DAE-BiLSTM model consists of a combination of Deep AutoEncoder (DAE) and Bidirectional Long Short-Term Memory (BiLSTM). BilSTM was used, which is one of the most important deep learning techniques that give the best results. By using BilSTM, the amount of network information available increases, which effectively improves the context available for the algorithm. To improve its results and reduce the evaluation time, DAE should be used alongside it. The importance of DAE is to compress the original data, remove the noise, extract the most important features, and eliminate redundant data. The proposed work also consists of several phases. In the data preprocessing phase, the dataset is optimized by removing irrelevant and redundant data, validating the data type, dealing with missing values, etc. Using different ML techniques to choose the important features, the best one is selected for this work to ensure the best performance in the next phase. The DAE phase receives preprocessed data and compresses this data into a low-dimensional representation to extract useful knowledge. The compressed data generated from the DAE is then used to train the BiLSTM classifier based on binary classification.

The main contribution of the paper is proposing an efficient Fog based IDS to accurately detect the vulnerabilities that penetrate fog devices to obtain sensitive data and violate fog services. The following is a summary of the contributions of the proposed IDS model:

- It proposes a combination of DAE and BiLSTM to enhance intrusion detection performance which requires less amount of dataset used.
- The DAE model is trained to compress the original data to further extract features and reduce complexity.
- The compressed representation of the original dataset is used to train the BiLSTM model to get a better performance in predicting the intrusion or not.
- The dataset is prepared, preprocessed, and optimized by comparing multiple techniques to select features and choosing the best one for the model.
- A comparative analysis between the proposed work and some related IDS works on the fog environment using the ToN_IoT and NSL-KDD datasets. The results of the experiments demonstrate that the proposed work performs better than other methods in terms of metrics like accuracy, precision, recall, and F1-score.

The following is how the paper is organized: the background knowledge of fog computing, some of the security problems it faces, and the role of IDS as an effective solution to these problems are presented in Section 2. Section 3 presents some work related to IDS based on ML and DL in a fog computing environment and its results and limitations. The DAE-BiLSTM model and some of its basic concepts are also explained for clarity in Section 4. In Section 5, the different experiments were discussed, their results were compared, their performance was determined, and there was a comparison with other methods already in use. In Section 6, the conclusion resulting from the proposed work is drawn and clarified, in addition to directions and improvements that we hope will benefit future work being presented.

2. BACKGROUND

In this section, more information will be covered about fog computing, some security issues, cyber-attacks on them, and how to use intrusion detection systems to solve them. First, a simplified view of fog computing and its importance is presented, followed by a list of security issues and malicious attacks that threaten IoT devices and fog. Finally, intrusion detection systems, their categories, and their importance in solving these problems will be discussed.

2.1 Fog Computing

Fog systems have the lower computing power for resources but can be scaled up as needed, such as memory, processing, and storage. Fog is a mini-cloud at the edge of the network that is closer to the

user to reduce latency and support real-time applications. The fog layer includes fog nodes such as access points, switches, embedded servers, controllers, routers, gateways, and storage devices. End-users interact with these nodes instead of connecting to cloud data centers to reduce power consumption and respond in real time. Different scenarios of fog nodes were presented in [10], where fog nodes were divided into two categories according to the edge devices connected to them. The first category is dumb fog nodes, which are limited to producing and sensing data only. The second category is smart fog nodes, which do not stop at sensation but also preprocess data using advanced computing capabilities. Fog computing offers load balancing, lowers latency, and maintains the quality of service (QoS), in addition to lowering expenses and energy consumption within the network [2].

2.2 Fog Security Issues

The fog platform is vulnerable to many attacks due to its presence between users and the cloud, which creates many vulnerabilities. Fog computing is currently used to increase the performance of websites. Fog computing enables them to deal with HTTP requests, receiving and executing them at the same time, as well as dealing with different user files [11]. This makes fog computing vulnerable to attacks based on malicious input without validation, such as Cross-Site Scripting (XSS) and injection attacks. Fog nodes communicate with each other and with a huge number of IoT devices. This makes them vulnerable to a network traffic interception attack called Man-In-The-Middle (MITM) by a malicious intermediary attacker who intercepts communications, alters data, and destroys service within the network [12]. As shown in Figure 1, the fog platform connects to various types of IoT devices for ordinary users, such as sensors, laptops, phones, etc. This connection can be both wired and wireless, which means the fog platform is easily accessible, making it vulnerable to resource violation attacks such as denial of service (DoS) and distributed denial of service (DDoS) [13]. Some attacks represent a great danger within the IoT and affect its performance and lead to damage, such as scanning attacks that aim to collect data (available system services and open ports) and backdoor attacks that use hidden malware to gain remote control of IoT systems. Using different techniques to hack the passwords of IoT devices is a common attack within the fog environment. Fog computing is vulnerable to ransomware attacks that prevent a user from gaining access to an IoT device or service [14]. The ToN_IoT dataset includes all of these cyber-attacks that threaten the fog layer, such as backdoors, ransomware, scanning, denial of service (DoS), password attacks, Man-In-The-Middle (MITM), distributed denial of service (DDoS), data injection, and Cross-site Scripting (XSS) [15].

2.3 Intrusion Detection System

The fog network is vulnerable to many intrusions. The intruder can eavesdrop, steal the password, spread malicious programs, malicious injections, or violate the resources of the network, and thus he can access another user's account without permission or destroy the entire network. Therefore, it is necessary to use advanced and different methods of protection to limit these attacks. There are many techniques and tools to protect the network from this intrusion, such as firewalls, encryption techniques, and intrusion detection systems (IDSs). This research will focus on IDS within the fog environment. IDS is a security system that is applied inside fog nodes and monitors data coming from different sources, such as IoT devices or neighboring fog nodes. IDS uses various advanced learning methods, such as machine learning (ML) and deep learning (DL) models, to detect undesirable and abnormal behavior within the network [4]. IDS is categorized into two types: Network Intrusion Detection Systems (NIDS) and Host Intrusion Detection Systems (HIDS) [16]. Network-based IDS monitors network traffic, while HIDS monitors and analyzes activities, logs, and modifications to system files to identify intrusions. IDS can be classified into two categories: anomaly-based IDS and misuse-based IDS or signature-based IDS [4]. NIDS monitors the behavior of fog network traffic through various IoT devices and searches for anomalies. If the behavior and patterns of current network traffic differ, it issues an alert. The structure of IDS consists of two methods: distributed or centralized. Distributed IDS nodes are scattered across the network, and each contributes to a portion of the processing with the others to form a collective data processing. Centerized IDS nodes process the network data entirely. Having NIDS inside the fog nodes helps to have a wider view of the network as the fog nodes receive different data from the edge subnetworks. IDS is constantly and rapidly being developed in fog computing as new attacks have emerged that need immediate solutions.

3. RELATED IDS WORKS

This section discusses the latest security techniques and some previous work in the fog

computing environment that uses ML and DL to detect attacks faced by IoT.

Kumar et al. [13] presented a design based on IDS technology in fog computing using an ensemble of different ML techniques. At the first level, K-Nearest Neighbors, XGBoost, and Gaussian Naive Bayes are combined as separate learners, and then use the results of the first level to predict the second level by Random Forest for classification. When it was used on the UNSW-NB15 dataset, the detection rates for backdoor attacks, analysis, reconnaissance, and DoS attacks were 71.18%, 68.98%, 92.25%, and 85.42% respectively. Their work gives an overall score of detection accuracy of 93.2%. The model is computationally complex and has low detection accuracy for the backdoor, analysis, and worm attacks.

Sudqi Khater et al. [17] proposed a lightweight IDS for a fog computing environment using a Multilayer Perceptron (MLP) model with a Raspberry Pi as a fog node. It was applied on a dataset (ADFA-WD) with 74% accuracy, 74% recall, and 74% F1-score and on a dataset (ADFA-LD) with 94% accuracy, 95% recall, and 92% F1-score for detecting various attacks. Their model still has to be improved in terms of computational efficiency and detection accuracy by using modern learning models. It cannot detect current IoT network attacks.

Prabavathy et al. [5] presented an intrusion detection model for a fog computing environment using an online sequential extreme learning machine (OS-ELM) to detect incoming attacks from the Internet of Things and implemented it on the NSL-KDD dataset. The authors reached a detection accuracy of 97.36% when implemented in fog computing. The dataset used was unbalanced and needed sophisticated preprocessing before use. Although all the features available in the dataset are used, it is unable to detect current or unknown network attacks.

Kalaivani et al. [6] proposed a deep learning-based intrusion detection model for use in fog layer devices that combines CNN with LSTM called ICNN-FCID to classify attacks. They also compared three activation functions (relu, sigmoid, and tanh), with relu obtaining the best accuracy of 96.5%, 85.2% for precision, 91.1% for recall, and 86.4% for F1-score. The model was evaluated using NSL-KDD and this evaluation resulted in an attack detection accuracy of 96.5%. This accuracy needs to be improved. They also had to evaluate their best model on new and different datasets that correspond to the fog layer to determine its performance for different types of cyber-attacks. The proposed method was only trained to identify basic network attacks.

Sadaf et al. [7] presented a method to detect abnormal and intrusive attacks based on deep learning using AutoEncoder and Isolation Forest techniques within a fog computing environment. They called it (Auto-IF) and it was verified using the NSL-KDD dataset with an accuracy rate of 95.4%,

Table 1: Existing Works of IDS at Fog Layer

Authors	Methods	Dataset	Accuracy	Limitations
Kumar et al [13]	Ensemble of KNN, XGBoost, Gaussian Naive Bayes, and Random Forest	UNSW-NB15	93.2%	The model is computationally complex and has low detection accuracy for backdoor, analysis, and worm attacks.
Sudqi Khater et al [17]	Multilayer Perceptron (MLP)	ADFA-WD	74%	The model's 94% attack detection accuracy has to be increased. It cannot detect current IoT network attacks.
		ADFA-LD	94%	
Prabavathy et al [5]	Online Sequential Extreme Learning Machine (OS-ELM)	NSL-KDD	97.36%	Although all the features available in the dataset are used, it is unable to detect current or unknown network attacks.
Kalaivani et al [6]	CNN with LSTM (ICNN-FCID)	NSL-KDD	96.5%	The proposed method was only trained to identify basic network attacks.
Sadaf et al [7]	Autoencoder and Isolation Forest (Auto-IF)	NSL-KDD	95.4%	Attack detection accuracy for the model is 95.4%, which needs to be improved. It cannot detect recent or unknown IoT network threats.
Maharani et al. [18]	K-Means	KDD Cup'99	93.3%	Attack detection accuracy for the model is 93.3%, which can be higher. It cannot detect recent or unknown IoT network threats.

94.8% for precision, 97.2% for recall, and 96% for F1-score for detecting different attacks in real-time. Although the isolation forest is used to increase the accuracy of their results, the accuracy of their final model still needs to be improved. It cannot detect recent or unknown IoT network threats. It also needs to be evaluated with recent datasets.

Maharani et al. [18] proposed an IDS framework for fog computing to detect attacks using ML techniques, namely Decision Tree, K-Means, and Random Forest. The evaluation was performed using the KDD Cup'99 dataset. It found that K-Means has the highest accuracy of 93.3% among other algorithms. KDD Cup'99 is one of the oldest and most outdated datasets, so it was preferable to evaluate this model with newer datasets. Their accuracy is small compared to the current works, so they should improve it by using new and effective learning algorithms. It cannot detect recent or unknown IoT network threats.

IDS has been used in all the above-mentioned works to scan network data packets and identify different types of attacks, but it still has some shortcomings that were discussed above that need to be worked on and improved. It was found that there are repeated shortcomings in most of these works, such as:

- As shown in Table 1, the accuracy rates of the used models need to be improved to ensure the efficiency of the intrusion detection system and reduce the high rate of false alarms.
- Most of the work was evaluated on older datasets that lack modern attack types in the fog layer. Therefore, it is not compatible with current Internet of Things devices.

In this paper, to overcome these shortcomings, an IDS model based on deep learning was implemented in the fog layer. A network-based IoT dataset called ToN_IoT is used that contains nine types of cyber-attacks that have been gathered from huge, realistically tested networks.

4. PROPOSED IDS ARCHITECTURE

As mentioned earlier, the network structure consists of three layers: edge, fog, and cloud. These layers collaborate in dealing with traffic from various devices to detect intrusion and record it in blacklists so that similar packages are not included again.

As shown in Figure 2, fog layer nodes receive IoT device packets. By building an intrusion detection system model inside these nodes using DL, this data is analyzed and the intrusion is detected. The fog node intrusion detection system collects traffic, generates security alerts if an intrusion is detected, then logs those alerts and sends them to the cloud servers.

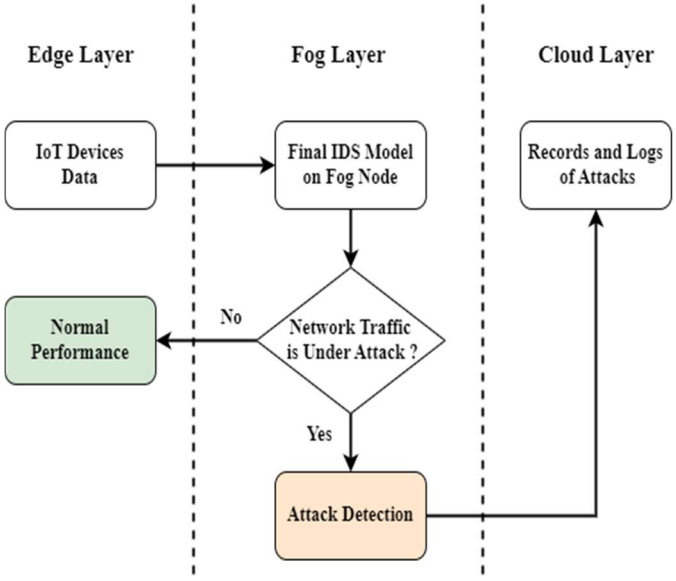

Figure 2: Flow-graph for the proposed IDS Architecture

IDSs must be enhanced to ensure effective intrusion detection and lower the high rate of false alarms as new generations of cyber-attacks appear every day. Enhancements are presented by utilizing advanced DL techniques to guarantee great results in identifying intrusions and a recent dataset to feed these technologies. The DAE-BiLSTM model includes several phases. The outputs of each phase are used as inputs to the next, as shown in Figure 3. The proposed IDS model has been tested on the network ToN_IoT dataset and consists of several phases; each phase has separate functions as follows:

Phase (1): Preprocessing techniques should be used that aim to improve and prepare the data before it is used by deep learning techniques. In the proposed model, preprocessing consists of:

- Data cleaning is done to get rid of messy and ineffective data and to ensure the correctness of the types of features.
- Feature selection for defining the most important features of data by using the appropriate method.
- Feature encoding encodes feature values into numerical values.
- Feature scaling for scaling feature values into specific ranges.

Phase (2): Deep AutoEncoder (DAE), which is also a good option for guaranteeing data privacy and further feature selection.

Phase (3): Bidirectional long short-term memory (BiLSTM), which is currently one of the most effective deep learning models employed. The use of BiLSTM has increased the accuracy of results and the ability to detect attacks effectively.

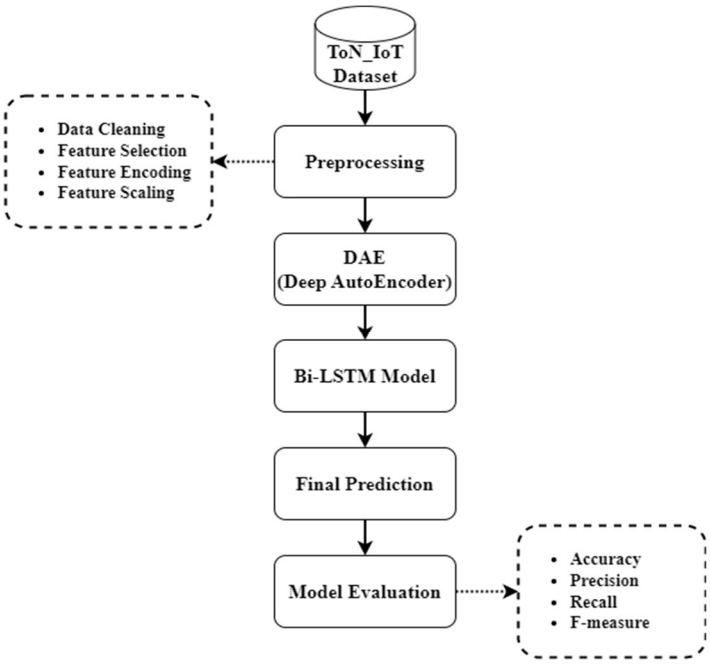

Figure 3: DAE-BiLSTM Model.

As shown in Figure 3, the DAE-BiLSTM model includes several stages. The outputs of each stage are used as inputs for the next. The ToN_IoT dataset is initially prepared and optimized by preprocessing techniques including data cleaning and segmentation, feature selection, feature encoding, and feature scaling. The output is used as input to DAE for further feature selection and encoding. The DAE output is then used as input into BiLSTM to increase the accuracy of the results and the ability to detect attacks efficiently.

The following is a brief explanation of the proposed IDS model, the methods used for each of its phases, and their advantages.

4.1 Data Preprocessing

Fog nodes exist between IoT devices and the cloud layer. They receive incoming traffic from IoT devices with various feature categories as numeric and categorical contents. To improve the efficiency of the DAE-BiLSTM model, the traffic must be analyzed and preprocessed as described below.

4.1.1 Data cleaning

Data cleaning is an important process for identifying bad data and systematically restructuring it correctly, as data must be cleaned before processing to ensure the quality of the results of the machine and deep learning models. There are many methods of data cleaning, such as removing irrelevant and redundant data, checking the correctness of the data type, handling missing values, etc.

- Remove Duplicate Rows and Check Data Types

The presence of duplicate data rows leads to data distortion, which then negatively affects the results, so it is preferable to remove them. It is also necessary to ensure the data type for each feature within the datasets, whether it is numeric, nominal, boolean, or other.

- Dealing with Missing Values

Lots of datasets currently in use have missing values. Ignoring missing values can be a mistake because it weakens your data, and you won't get accurate results. These values can be dealt with in two ways: either by removing them or by inputting them.

Removing the missing value is a sensitive process as it either benefits the data set or leads to the removal of useful insights from your data. You must verify and analyze these values before removing them. The missing values should not be ignored and must be filled in. They are analyzed according to the nature of the dataset used.

4.1.2 Feature selection

One of the most crucial steps is feature selection, as the best features have to be chosen from many features. This process removes redundant and irrelevant features to reduce computation time, improve accuracy, and build a DL model that achieves higher performance. In the proposed model, Random Forest Classifier, Chi-square test, and Extra-Trees are applied for feature selection.

- Chi-square Test

A statistical test known as the Chi-square test [19] that belongs to the filter method category can be effectively used to select important features. It is applied to a set of features to evaluate the correlation between them by calculating the Chi-square

statistical value of the feature concerning the output class. The Chi-square formula is applied using equation (1).

$$x^2 = \sum (O_i - E_i)^2 / E_i \quad (1)$$

Where O_i represents the observed value and E_i represents the expected value.

- Random Forest Classifier

Using the Random Forest Classifier to select features falls under the category of embedded methods that combine the properties of filter and wrapper methods [20]. The algorithm's work depends on classification and regression trees (CART). Each tree in the random forest calculates the importance of the feature according to the purity of its leaves. There is a direct relationship between the increase in the purity of leaves and the increase in the importance of the feature. The average is calculated for all trees, and this average is normalized to 1.

- Extra-Trees

The extra-trees classifier is an embedded method to extract relevant features [21]. They generate from the training dataset a large number of individual decision trees. The algorithm chooses a random split rule from the sub-features and a random cut point relative to the root node. The parent node is randomly divided into two child nodes, and this process is repeated until the leaf node has no child nodes. A majority vote of all the trees' predictions determines the final prediction. The Gini importance of each feature during forest construction is calculated and arranged in descending order of importance. The best features are chosen as an introduction to the different learning models.

The chi-square is used in the model because of its many advantages. It is considered robust and flexible in dealing with data. It was also found that it is distinguished by its ease and faster computations, in addition to the specific information that can be obtained from the test [19]. When applying the methods, it was found that using the chi-square method led to higher results than other methods.

4.1.3 Feature encoding

This step is one of the most important processes for dealing with data, as categorical data is encoded into numeric data because dealing with numbers is generally easier and faster. Categorical data can be encoded in a variety of ways, including label encoding and one-hot encoding. Label encoding is used because there is a lot of categorical data in the dataset. Using label encoding makes it fast and easy and does not create a messy frame of data as in one-hot encoding, which adds a lot of columns.

4.1.4 Feature scaling

The values of the features in the network traffic coming from IoT devices vary, some of them have very high values, while others have very small values, so there is a need to use feature scaling. Feature scaling is one of the most important steps during data preprocessing. Scaling can make a difference during the creation of a deep learning model as it determines whether a model is weak or strong. The most common scaling techniques are normalization and standardization. The Min-Max scaling [22] is used, to apply normalization as it scales and translates each feature individually so that it is within the specified range. It can be calculated as follows:

$$X_{sc} = (X - X_{min})/(X_{max} - X_{min}) \quad (2)$$

Where X_{sc} represents the output of scaling, X represents the input value, X_{min} represents the minimum value, and X_{max} represents the maximum value.

4.2 DAE-BiLSTM Model

The deep learning methods used in the DAE-BiLSTM model are discussed in this section. These methods were tested using the enhanced dataset obtained through preprocessing. The DAE-BiLSTM model consists of DAE and BiLSTM.

4.2.1 AutoEncoder (AE)

One of the types of neural networks, called AutoEncoder (AE) [23], is used for feature extraction. AE learns a compressed representation of raw data and then uses it to train a different deep-learning model. There are many types of AEs nowadays, such as deep AE, sparse AE, variational AE, denoising AE, etc. [23]. DAE is widely used for compressing data, reducing the feature dimensions, extracting relevant features, and removing noise. It aids in the selection of the most important features because it employs reduced encoding as a representative of the original data. The DAE consists of an encoder, bottleneck, and decoder, as shown in Figure 4. The encoder compresses the

input, while the decoder attempts to rebuild from the encoder's compressed version. The performance of the DAE and its effectiveness is evaluated by reconstruction loss. Reconstruction loss is done by evaluating the overall performance of the decoder and measuring the similarity between its output and the initial input. The reconstruction loss is estimated using the mean square error (MSE) method. DAE is used because it is used for datasets with real data values and because its encoding layer is compact and fast.

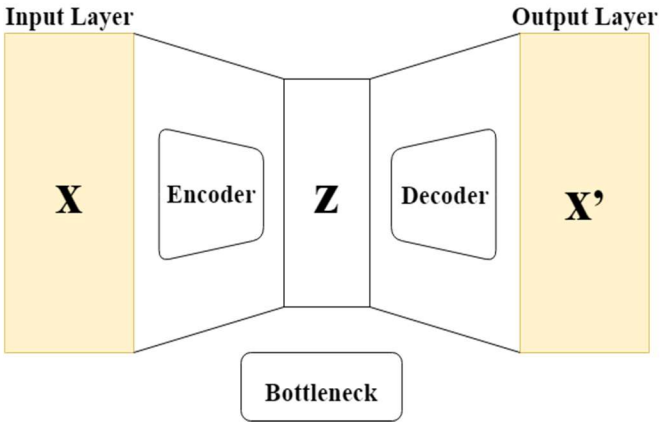

Figure 4: AutoEncoder

$$Z = E(X) \quad (3)$$

$$X' = D(Z) \quad (4)$$

Where E represents the encoder function, X represents the input value, D represents the decoder function, Z represents the compressed value, and X′ represents the output value.

4.2.2 LSTM

One of the most important types of recurrent neural networks (RNN) is called long short-term memory (LSTM), which can learn from long dependencies and was introduced in 1997 [24]. LSTM can be an effective solution for the fog environment compared to other deep learning models. It is based on learning from long sequences by storing information about the network's previous state using the forget gate. LSTM can use the historical data stored from previous network traffic for a while [25]. The LSTM cell [24] contains a set of basic gates, namely input gates, forget gates, and output gates, and their function is to control cell states and their protection. LSTM reduces the long time it takes to detect attacks compared to other learning models such as DoS and DDoS, which are major threats to the fog network. The data generated by IoT devices is unstructured and has various forms. LSTM can effectively learn from unstructured data and extract effective insight. LSTM performance increases with more training data, and these conditions lead to LSTM's superiority over other deep learning techniques.

- BiLSTM

Bidirectional LSTM (BiLSTM) [26] is a sequence processing model derived from the LSTM model, which consists of two LSTMs as shown in Figure 5, one of which takes the input in a forward direction while the other takes a backward direction Input layer, forward transmission layer, reverse transmission layer, and output layer are the four layers into which BiLSTM is divided [27]. BiLSTM has been developed for speech recognition and handwriting recognition, where one of its most important benefits is that it has an input sequence based on past and future sequences. BiLSTM increases the amount of network information available, which effectively improves the context available for the algorithm. It was found that, although it is expensive, training the data using BiLSTM models shows better predictions than LSTM models in the normal mode.

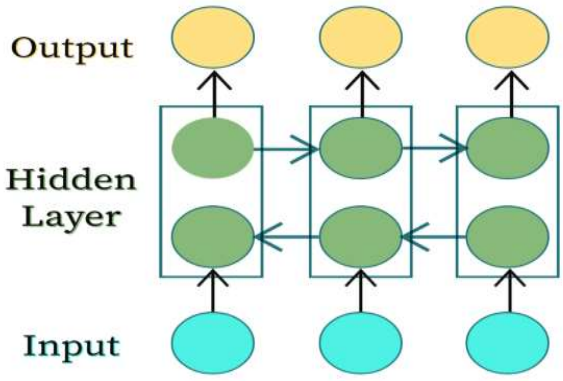

Figure 5: BiLSTM Architecture

5. EXPERIMENT RESULTS AND DISCUSSION

This section presents the DAE-BiLSTM model through different experiments, analysis of its performance, and comparison with the latest models currently used in the same environment. Experiments were conducted in Python using a Jupyter notebook on an Anaconda machine. For DAE, packages from Keras and Tensorflow are

used. For the BiLSTM method, packages from Keras are used. The ToN_IoT and NSL-KDD datasets are used to evaluate the performance of the DAE-BiLSTM model. These experiments were performed on a machine running 64-bit Windows 10 with an Intel (R) Core i5-8265U CPU @ 1.80 GHz and 12 GB of memory.

5.1 Datasets

5.1.1 Common existing datasets

As mentioned earlier in the related IDS work in this research, one of the most important datasets used in IDS at the fog layer is the NSL-KDD dataset. It is used to identify attacks like DoS, SQL injection, and brute force [28]. It contains 41 numerical and nominal features [8]. Currently, NSL-KDD has some limitations and is an outdated and widely used dataset for network intrusion detection. KDD Cup'99 [29], UNSWNB15 [30], and CICIDS [31], like NSL-KDD, are outdated with existing networks and not sufficient to represent the normal state of fog-based IoT networks that include many different standards [32]. There are also other shortcomings, such as the fact that datasets do not contain heterogeneous data, such as DEFCON and LBNL datasets. Other new datasets have been adopted that are in line with the nature of IoT and depend on the new network architecture, such as ToN_IoT, Aposemat IoT-23 [33], and N-BaIoT [34].

5.1.2 ToN_IoT dataset

One of the most important datasets currently is the ToN_IoT dataset. It represents new generations of IoT, industrial IoT, operating systems, and network traffic datasets. ToN_IoT datasets have been gathered from huge testbed networks realistically with a testbed design that has three layers: edge/IoT, fog, and cloud [35]. They include raw and processed data sources gathered from Windows and Linux operating system datasets, network traffic datasets, and IoT service telemetry datasets [9]. The ToN_IoT network traffic dataset is used for network intrusion detection to evaluate several AI-based cyber security solutions. They are not limited to general network features, but also include a set of services to aid in the detection of attacks, such as FTP, DNS, and HTTP. The ToN_IoT dataset contains hundreds of records. Each file contains 45 attributes categorized into four groups that include connection, statistics, user, and violation attributes [36]. Among the ToN_IoT datasets, the ToN_IoT network dataset was used since it contains traffic data from IoT devices connected to the fog network. There are 46 features in the ToN_IoT network dataset; two of them are class label and type which is either normal or attacked. The "Train_Test_Network" file, a simplified sample file, was used.

It was used to evaluate the effectiveness of current cyber-security solutions using different AI methods. There are 300,000 records of normal network traffic and 161,043 records of attacks in this dataset, totaling 461,043 records. Table 2 shows the number of network traffic records for each type in the dataset.

Table 2: Number of Records for Each Type of Network-based train-test Dataset of ToN_IoT

Type	Numbers of records
normal	300000
backdoor	20000
ransomware	20000
DoS	20000
DDoS	20000
injection	20000
XSS	20000
password	20000
scanning	20000
MITM	1043

Table 3 shows different types of cyber-attacks in the ToN_IoT dataset that threaten the fog layer, such as: backdoor, ransomware, scanning, denial of service (DoS), password attack, Man-In-The-Middle (MITM), distributed denial of service (DDoS), data injection, and Cross-site Scripting (XSS).

Table 3: ToN_IoT Attack Types

Attack Category	Attack Type	Fog network vulnerabilities
Flooding Attacks	DoS	The normal traffic of fog servers is overwhelmed by a flood of data packets, making them inaccessible.
	DDoS	
Injection Attacks	XSS	Sending malicious scripts to fog and IoT devices, through which sensitive information can be obtained.
	Data Injection	Injecting input for reading, modifying, and deleting sensitive data inside the fog devices.
Information Gathering	Scanning	Identifying fog network security vulnerabilities.

Malware attacks	Backdoor	Control components of the fog network by installing backdoors.
	Password	Identifying a password to a fog device to obtain unauthorized access to fog resources.
	Ransomware	Encrypting data and systems of fog devices to pay a fee to the attacker.
Eavesdropping	MITM	Communication interception between IoT devices and a server such as DNS and ARP spoofing.

5.1.3 Experiment scenarios

- Preprocessing Process

Preprocessing datasets helps learn models to save time and reduce computation while ensuring increased model performance. The ToN_IoT dataset is new, with new attacks and feature values not found in previous datasets. So data problems within it had to be worked on, such as removing irrelevant and redundant data, validating data types, handling missing values, etc. The ToN_IoT dataset contains records with features that may be nominal, binary, or numeric. The stages of the preprocessing phase were applied as follows:

○ Data Cleaning

These steps are taken to prepare and clean the ToN_IoT network dataset:

1) The dataset was modified by removing the redundant rows.
2) Null values are expressed within the data set with the symbol "-". The used "-" symbol has been replaced with an actual null value to help us with the next process.
3) The validity of each data type for each feature within the dataset, whether numeric, nominal, or boolean, was also verified.
4) The missing categorical values are replaced by a category called "other" and the missing numeric values are replaced by 0.
5) Some features with null values above 99% have been removed as being unimportant to the dataset.
6) The features of IP addresses and ports have also been removed based on the recommendation found in [35] when using new learning models to increase detection efficiency and reduce the false alarm rate.

Table 4 shows the new number of network traffic records for each type in the dataset after the data cleaning process.

Table 4: Number of Records for Each Type of the Dataset after Data Cleaning Process

Type	Numbers of records
normal	259962
backdoor	17984
ransomware	17972
DoS	16417
DDoS	18002
injection	17973
XSS	17947
password	17972
scanning	17993
MITM	961

○ Feature Selection

Random Forest Classifier, Chi-square Test, and Extra-Trees are applied for feature selection to reduce the number of features and identify the most important ones. The chi-square test is used because it is robust and flexible in dealing with data. Furthermore, it is distinguished by being faster and easier to calculate. When applying the Chi-square Test, Random Forest Classifier, and Extra-Trees, it was found that using the Chi-square method to select the most important features as part of the proposed model led to higher results in accuracy than the rest of the applied methods, as shown in Figure 6.

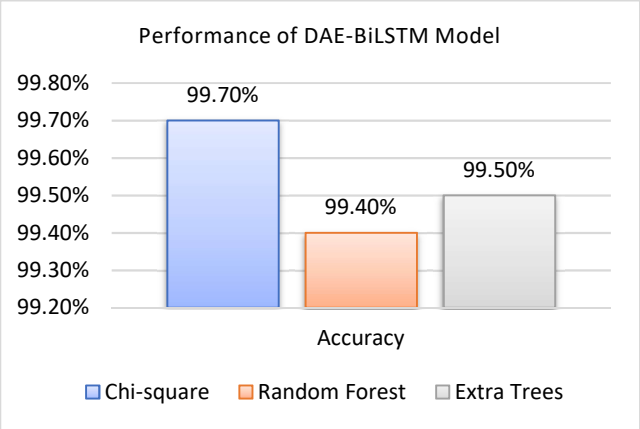

Figure 6: Performance of the DAE-BiLSTM Model using ToN_IoT Dataset with Different Feature Selection Techniques

The most important features are selected using the Chi-square test according to their score. The top

15 highest-scoring features in the dataset are selected. Table 5 presents the features selected in the ToN_IoT dataset using the Chi-square Test in the proposed model.

Table 5: Features Selected from ToN_IoT Dataset Using Chi-square Test

No.	Selected Features	Feature Description
1	proto	Flow traffic protocols at the transport layer
2	service	Other additional protocols found
3	conn_state	Connection states
4	duration	Flow duration
5	src_bytes	Source bytes number
6	dst_bytes	Destination bytes number
7	missed_bytes	Missing bytes number
8	src_pkts	Number of source packets
9	src_ip_bytes	Number of source IP bytes
10	dst_pkts	Number of destination packets
11	dst_ip_bytes	Number of destination IP bytes
12	dns_qclass	DNS query classes
13	dns_qtype	DNS query types
14	http_response_body_len	Transferred data content sizes from the HTTP server
15	http_status_code	Status codes of HTTP server

○ Feature Encoding

Deep Learning models in general, and DAE in particular, can only work with numeric data, so features must be encoded for numeric. The label encoder method is used for encoding because there is a lot of categorical data in the dataset. Using a label encoder makes it fast and easy and does not create a messy frame of data. The label encoder converts all categorical or nominal values of features into an integer to be added to their index. For example, the "proto" feature, which expresses the transport layer protocols of flow connections, is converted from categorical values (tcp, udp, icmp) to numeric values (1, 2, 3).

○ Feature Scaling

The remaining features after the label encoding stage need to be scaled and translated. Min-Max scaling functions are used for normalization. Each feature value is individually scaled into the specified range between 0 and 1 in the dataset. The resulting dataset is then split into train and test, with the train comprising 70% of the dataset and the test comprising 30% of the dataset. As a result, the DAE receives a set of preprocessed features.

- DAE Process

The encoder is defined as having three hidden layers to avoid overfitting and increase efficiency. The first layer has twice the number of inputs, the second has one and a half number of inputs, the third has the same number of inputs, and the bottleneck layer has half the number of inputs. Batch normalization and dropout ReLU activation are used to increase model learning efficiency. The decoder has three hidden layers, the first with the number of inputs, the second with the number of one and a half inputs, and the third with twice the number of inputs. DAE was applied with 100 epochs, 32-batch size, Adam optimizer, ReLU activation, and MSE loss. Figure 7 illustrates the difference between the reconstruction loss for the train and the test when applied to the ToN_IoT dataset. The output of the DAE process is reshaped to be suitable as a process input for BiLSTM.

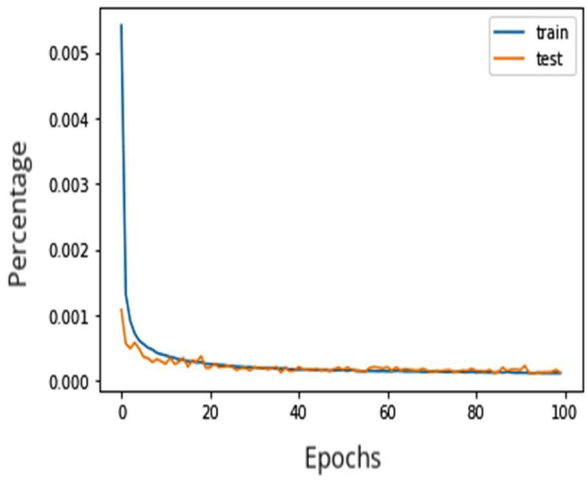

Figure 7: Reconstruction Loss of Train and Test Using ToN_IoT Dataset

- BiLSTM Process

Some different scenarios have been tested on BiLSTM to ensure high efficiency and accuracy. The model was implemented using a different number of BiLSTM layers in addition to changing the values of cells used for learning in each layer to choose the appropriate model for work. All scenarios with different numbers of layers were compared to choose the model with the highest efficiency and the fewest false alarms. It has been observed that as the number of layers increases, it

requires more learning time, and the accuracy decreases, which is not useful for this work. The proposed model consists of two BiLSTM layers, each with 32 cells, and one output layer with one cell using a sigmoid. It gives the highest accuracy for training and overcomes overfitting between training and testing. To overcome overfitting, dropout layers of 0.2 were also added. BiLSTM is applied with 50 epochs, the Adam optimizer, and ReLU activation in each layer except the last layer with sigmoid and binary_crossentropy loss. In terms of performance for binary classification, Figure 8 illustrates the BiLSTM's accuracy for both training and testing of the ToN_IoT dataset. It shows the increase in the train and the test accuracy with the increase in epochs to reach the best accuracy after 50 epochs with a 32-batch size. It also shows that the maximum training accuracy is 99.82% and the maximum testing accuracy is 99.87%.

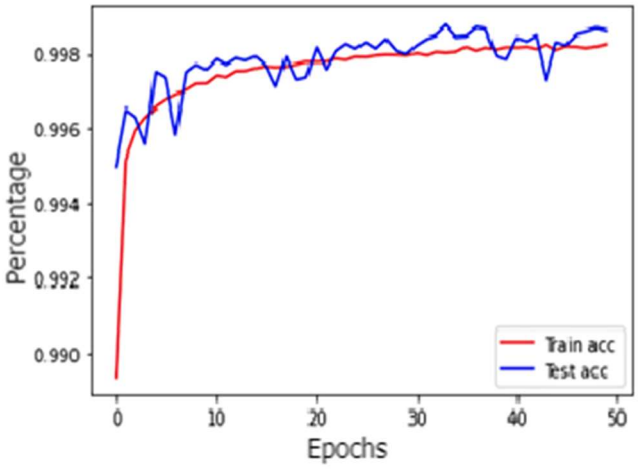

Figure 8: Accuracy for Binary Classification Using ToN_IoT Dataset

5.1.4 Evaluation metrics

The DAE-BiLSTM model has been validated on the new ToN_IoT dataset that is suitable for the fog computing environment. Then it was applied to the NSL-KDD dataset and compared to other models applied to the same dataset. This work achieves superior and very high results compared to others. Performance was evaluated by some metrics:

- Accuracy (Acc)

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

- Precision (P)

$$P = \frac{TP}{TP+FP} \quad (6)$$

- Recall (R)

$$R = \frac{TP}{TP+FN} \quad (7)$$

- F1-score

$$F1 = \frac{2*Precision*Recall}{Precision+Recall} \quad (8)$$

Where TP (True Positive): Only when the attack class is correctly predicted.
FP (False Positive): Only when the attack class is incorrectly predicted.
FP (True Negative): Cases in which the normal class is correctly predicted.
FN (False Negative): Only when the normal class is predicted incorrectly.

The performance of the DAE-BiLSTM model was evaluated on all datasets using the above-mentioned measures. On the ToN_IoT dataset, the DAE-BiLSTM model was tested, and the results were high compared to other models previously used, with high accuracy rates of 99.7%, 99.4% for precision, 99.7% for recall, and 99.5% for the F1-score. As shown in Figure 9, a comparison of the results of the proposed model with and without DAE is presented. From the results, we find an increase in all evaluation measures used. The DAE-BiLSTM model's evaluation time is 7.57% shorter than the model without DAE. Therefore, using a combination of DAE and BiLSTM improves the detection performance of malicious attacks.

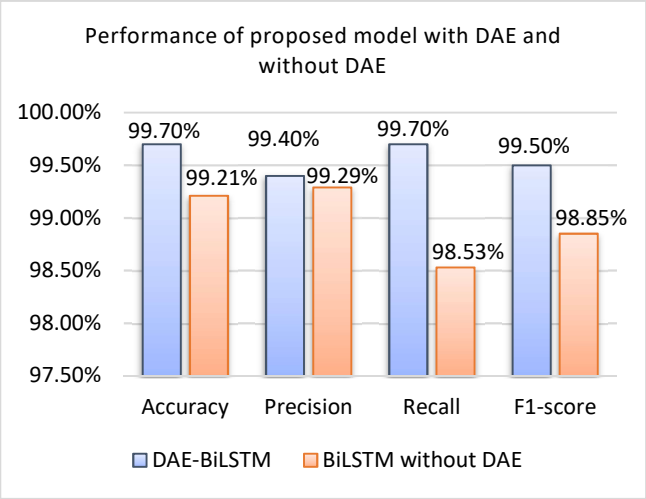

Figure 9: Performance of the DAE-BiLSTM Model Using the ToN_IoT Dataset

Table 6 compares the DAE-BiLSTM model's performance on the ToN_IoT dataset and the NSL-KDD dataset, along with the detailed accuracy, precision, recall, and F1-score outcomes for each dataset. Given the results, using the ToN_IoT dataset resulted in higher results compared to using the NSL-KDD dataset in the model, which reinforces the idea of using the ToN_IoT dataset. In addition, the ToN_IoT dataset is recent and more appropriate to the nature of IoT, based on the network hierarchy that includes the fog layer.

Table 6: Performance of the DAE-BiLSTM Model Using Different Datasets

Dataset	Accuracy	Precision	Recall	F1-score
ToN_IoT	99.7%	99.4%	99.7%	99.5%
NSL-KDD	97.7%	97.6%	96.9%	97.1%

Table 7 compares the accuracy of the DAE-BiLSTM model to the accuracy of other state-of-the-art works. These works were chosen for comparison because they use IDSs in the same fog layer and have previously been mentioned in related works. The DAE-BiLSTM model was applied using the NSL-KDD dataset to compare our results with the results of previous works [5], [6], and [7] of IDSs in the fog layer. In comparison to these works, our model achieved a high accuracy of 97.7% by using previously demonstrated preprocessing techniques as well as using DAE to extract important features and reduce the evaluation time for training the BiLSTM model. Because of the development and emergence of new attacks daily, it was necessary to use the ToN_IoT dataset. The ToN_IoT dataset is compatible with the nature of the fog layer and its interactions with each other and with IoT devices. The ToN_IoT dataset was prepared and preprocessed before applying the DAE-BiLSTM model. Achieved high results of 99.7% and high efficiency in detecting cyber-attacks. The results of the research [9] have been added to the comparison table. Although these results do not apply to IDSs on fog nodes, they apply different learning techniques to the same new ToN_IoT dataset used in the proposed work. Compared to all of these works, high results were obtained using the modern ToN_IoT dataset, which is more suitable for fog computing, and also using the NSL-KDD dataset.

Table 7: Comparison of the DAE-BiLSTM Model with the Previous Methods Used Before

Authors	Methods	Dataset	Accuracy
Kumar et al [13]	Ensemble of KNN, XGBoost, Gaussian Naive Bayes, and Random Forest	UNSW-NB15	93.2%
Sadaf et al [7]	Autoencoder and Isolation Forest (Auto-IF)	NSL-KDD	95.4%
Sudqi Khater et al [17]	Multilayer Perceptron (MLP)	ADFA-WD	74%
		ADFA-LD	94%
Prabavathy et al [5]	Online Sequential Extreme Learning Machine (OS-ELM)	NSL-KDD	97.36%
Kalaivani et al [6]	CNN with LSTM (ICNN-FCID)	NSL-KDD	96.5%
Booij et al [9]	Multilayer Perceptron (MLP)	ToN_IoT	97.8%
	Gradient Boosting Machine (GBM)		94.6%
	Random Forest (RF)		98%
DAE-BiLSTM model	**Deep AutoEncoder (DAE) and Bidirectional Long Short-Term Memory (BiLSTM)**	**ToN_IoT**	**99.7%**
		NSL-KDD	**97.7%**

6. CONCLUSION AND FUTURE WORK

Fog computing is used to handle incoming network traffic in real-time. Intrusion Detection System (IDS) is one of the most important ways to identify families of new attacks. An intrusion detection model based on deep learning, called DAE-BiLSTM, was presented to the fog layer for IoT devices. To improve performance and effectively detect cyber-attacks on the fog layer, advanced DL techniques were used. The proposed IDS model

consists of a combination of two different DL techniques. First, DAE is used to accurately select additional features without losing information and to reduce complexity. Second, the BiLSTM model is used to avoid penetrating fog devices by getting high accuracy in attack detection. Different test scenarios are performed on different datasets. Many current works use the NSL-KDD dataset, which is outdated and lacks new IoT cyber-attacks. In this context, the network-based ToN_IoT dataset is new and more suitable for IoT. It includes nine new types of cyber-attacks that threaten the fog layer. With a high accuracy rate of 99.7%, 99.4% for precision, 99.7% for recall, and 99.5% for the F1-score, the model was validated on the network-based ToN_IoT dataset to ensure avoiding violating fog services. This model was then validated using the widely used NSL-KDD dataset, and in comparison to other works used, it achieved a high accuracy rate of 97.7%%, 97.6% for precision, 96.9% for recall, and 97.1% for the F1-score. Future improvements to this work will come from applying more advanced deep learning techniques such as generative adversarial networks (GANs). Additionally, we plan to highlight the efficiency of the work by comparing many datasets in order to obtain the best results while taking into account the real-time handling of the IoT devices that are currently in use.

REFERENCES:

[1] Ometov, A., Molua, O. L., Komarov, M., & Nurmi, J., "A survey of security in cloud, edge, and fog computing", *Sensors*, vol. 22, 2022, p. 927.

[2] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues", *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, Apr 2015, pp. 2991–3005.

[3] N. Moustafa, "A systemic IoT–fog–cloud architecture for big-data analytics and cyber security systems: a review of fog computing", *Secure Edge Computing*, 2021, pp. 41–50.

[4] Mohamed, Rehab Hosny, Faried Ali Mosa, and Rowayda A. Sadek, "Efficient Intrusion Detection System for IoT Environment", *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022.

[5] S. Prabavathy, K. Sundarakantham, and S. M. Shalinie, "Design of cognitive fog computing for intrusion detection in Internet of Things", *Journal of Communications and Networks,* vol. 20, no. 3, 2018, pp. 291–298.

[6] K. Kalaivani and M. Chinnadurai, "A Hybrid Deep Learning Intrusion Detection Model for Fog Computing Environment", *INTELLIGENT AUTOMATION AND SOFT COMPUTING,* vol. 30, no. 1, 2021, pp. 1–15.

[7] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing", *IEEE Access,* vol. 8, 2020, pp. 167059–167068.

[8] Eshak Magdy, M., M MATTER, A. H. M. E. D., HUSSIN, S., HASSAN, D., & Elsaid, S, "A Comparative study of intrusion detection systems applied to NSL-KDD Dataset", *The Egyptian International Journal of Engineering Sciences and Technology*, 2022.

[9] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. den Hartog, "ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets", *IEEE Internet of Things Journal,* vol. 9, no. 1, 2021, pp. 485–496.

[10] E. M. Tordera, X. Masip-Bruin, J. Garcia-Alminana, A. Jukan, G. J. Ren, J. Zhu, and J. Farre, "What is a fog node a tutorial on current concepts towards a common definition", *arXiv preprint arXiv:1611.09193,* 2016.

[11] Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R., & Madinabeitia, G, "On the detection capabilities of signature-based Intrusion Detection Systems in the context of web attacks", *Applied Sciences*, vol. 12, no. 2, 2022, p. 852.

[12] A. Mallik, "Man-in-the-middle-attack: Understanding in simple words", *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, vol. 2, no. 2, 2019, pp. 109-134.

[13] P. Kumar, G. P. Gupta, and R. Tripathi, "A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks", *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, 2021, pp. 9555–9572.

[14] D. Grzonka, A. Jakobik, J. Kołodziej, and S. Pllana, "Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security", *Future generation computer systems*, vol. 86, 2018, pp. 1106–1117.

[15] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets", *Sustainable Cities and Society*, vol. 72, 2021, p. 102994.

[16] Gamal, M, Abbas, HM, Moustafa, N, Sitnikova, E & Sadek, RA, "Few-Shot Learning for Discovering Anomalous Behaviors in Edge Networks", *Computers, Materials and Continua,* vol. 69, no. 2, 2021, pp. 1823-1837.

[17] B. Sudqi Khater, A. W. B. Abdul Wahab, M. Y. I. B. Idris, M. Abdulla Hussain, and A. Ahmed Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing", *applied sciences*, vol. 9, no. 1, 2019, p. 178.

[18] Maharani, M. P., Daely, P. T., Lee, J. M., & Kim, D. S, "Attack detection in fog layer for IIoT based on machine learning approach", *2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE*, 2020, pp. 1880-1882.

[19] M. L. McHugh, "The chi-square test of independence", *Biochemia medica*, vol. 23, no. 2, 2013, pp. 143–149.

[20] Christo, V. E., Nehemiah, H. K., Brighty, J., & Kannan, A, "Feature selection and instance selection from clinical datasets using co-operative co-evolution and classification using random forest", *IETE Journal of Research*, vol. 68, no. 4, 2022, pp. 2508-2521.

[21] Kharwar, Ankit Rajeshkumar, and Devendra V. Thakor, "An ensemble approach for feature selection and classification in intrusion detection using extra-tree algorithm", *International Journal of Information Security and Privacy (IJISP)*, vol. 16, no. 1, 2022, pp. 1-21.

[22] A. Pandey and A. Jain, "Comparative analysis of KNN algorithm using various normalization techniques", *International Journal of Computer Network and Information Security*, vol. 9, no. 11, 2017, p. 36.

[23] S. Zavrak and M. ˙ Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder", *IEEE Access*, vol. 8, 2020, pp. 108 346–108 358.

[24] Kumar, Ajay, and Amita Rani, "LSTM-Based IDS System for Security of IoT", *Advances in Micro-Electronics, Embedded Systems and IoT. Springer. Singapore*, 2022, pp. 377-390.

[25] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using LSTM networks", *Computers in Industry*, vol. 131, 2021, pp. 103498.

[26] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging", *arXiv preprint arXiv:1508.01991*, 2015.

[27] Z. Fu, "Computer network intrusion anomaly detection with recurrent neural network", *Mobile Information Systems*, 2022.

[28] M. S. Al-Daweri, K. A. Zainol Ariffin, S. Abdullah, and M. F. E. Md. Senan, "An analysis of the KDD99 and UNSW-NB15 datasets for the intrusion detection system", *Symmetry*, vol. 12, no. 10, 2020, p. 1666.

[29] Yusof, Nur Nadiah Mohd, and Noor Suhana Sulaiman, "Cyber attack detection dataset: A review", *Journal of Physics: Conference Series*, Vol. 2319, no. 1. IOP Publishing, 2022.

[30] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)", *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[31] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset", *Journal of Physics: Conference Series*, vol. 1192, no. 1. IOP Publishing, 2019, p. 012018.

[32] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques", *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, 2019, pp. 2671–2701.

[33] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Using MedBIoT Dataset to Build Effective Machine Learning-Based IoT Botnet Detection Systems", *International Conference on Information Systems Security and Privacy. Springer*, 2020, pp. 222–243.

[34] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders", *IEEE Pervasive Computing*, vol. 17, no. 3, 2018, pp. 12–22.

[35] N. Moustafa, M. Ahmed, and S. Ahmed, "Data analytics-enabled intrusion detection: Evaluations of ToN_IoT linux datasets", *2020 IEEE 19th International Conference on Trust, Security and Privacy. Computing and Communications (TrustCom). IEEE,* 2020, pp. 727–735.